

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

Appl. No.	:	09/884741
Applicant	:	David Holzer
Filed	:	06/18/2001
TC/A.U.	:	2143
Examiner	:	David E England
Confirmation No.	:	4756
Docket No.	:	Digi-P007
Customer No.	:	40418
Title	:	Method and Apparatus for Relationship Management

COMMUNICATED VIA EFS TO USPTO on 08/03/2010

APPELLANT'S APPEAL BRIEF Under 37 C.F.R. § 41.37

Dear Sir:

Applicant (Appellant) hereby submits this Appeal Brief pursuant to 37 C.F.R. § 41.37 in connection with the above-referenced application and respectfully requests consideration by the Board of Patent Appeals and Interferences for allowance from a rejection decision by the Examiner. The Examiner's rejection decision ("Office Action" or "Office") was communicated on 06/30/2009 and rejected all claims (1-30). Applicant has submitted the fees for filing an Appeal Brief required by 37 C.F.R. § 41.20(b)(2). Applicant submitted a Notice of Appeal that was received on 01/04/2010.

TABLE OF CONTENTS

	<u>Page</u>
I. REAL PARTY IN INTEREST	3-3
II. RELATED APPEALS AND INTERFERENCES.....	3-3
III. STATUS OF CLAIMS	3-3
IV. STATUS OF AMENDMENTS.....	3-4
V. SUMMARY OF THE CLAIMED SUBJECT MATTER	5-8
VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL.....	9-9
VII. ARGUMENT	10-24
Claim 27 Rejection under 35 U.S.C. § 102(e)	14-14
Claim 19 Rejection under 35 U.S.C. § 102(e)	15-15
Claim 1 Rejection under 35 U.S.C. § 102(e)	16-18
Claim 2 Rejection under 35 U.S.C. § 102(e)	18-19
Claim 13 Rejection under 35 U.S.C. § 102(e)	19-23
Conclusion	24
VIII. CLAIMS APPENDIX.....	25-27
IX. EVIDENCE APPENDIX.....	28-1433
(A) Evidence for Claims 1-22 – Relied Upon.....	28-43
(B) Evidence for Claims 1-22 – Entered by Examiner.....	44-1203
(C) Evidence submitted by Applicant	1204-1432
X. RELATED PROCEEDINGS	1433

I. REAL PARTY IN INTEREST

The real party in interest of Appellant is Digi International, Incorporated.

//

II. RELATED APPEALS AND INTERFERENCES

Appellant is unaware of any related appeals or interferences.

1) There are no prior or pending interferences.

2) There are no pending appeals before the USPTO.

3) There is no judicial proceeding related to Appellant's instant application or related applications.

//

III. STATUS OF CLAIMS

Claims 1, 2, 13, 19, 27: rejected

Claims 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30: canceled conditioned on this Appeal going before BPAI and a decision rendered on the claims being appealed, otherwise they are not canceled (e.g. the Examiner decides to yank the Application back into prosecution).

Claims 1, 2, 13, 19, 27: being appealed

//

IV. STATUS OF AMENDMENTS

Amendments in compliance with 37 CFR 1.116(b)(1) are made herewith without prejudice as indicated in **STATUS OF CLAIMS** regarding conditionally canceled claims to reduce the matters which must be considered by the Board of Appeals. Should this Appeal not proceed to a decision rendered by the Board on the Appealed claims then Amendments in compliance with 37 CFR 1.116(b)(1) are rescinded.

//

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

Appellant has indicated below representative Figures/Specifications. Appellant has not listed all figures or places in the specification. If the Office declares this unacceptable, Applicant reserves the right to amend the Appeal.

27. A method of relationship management comprising:

receiving information from a device automatically when the device is connected to a network; **{Figure 5, Figure 6 at 602; Specification: page 14 line 1 – page 16 line 2}**

determining if the device has access rights to a service provider after said receiving information from said device; and **{Figure 6 at 604; Specification: page 15 lines 16-21}**

if the device has access rights to the service provider, sending information to the device on how to contact the service provider. **{Figure 6 at 608; Specification: page 15 lines 16-21}**

19. A system comprising:

a plurality of service providers in communication with a network; **{Figure 5 at 506-1 through 506-M, 504; Specification: page 14 line 1 – page 15 line 15}**

a plurality of devices sending communications automatically with information about themselves when connected to the network; and **{Figure 5 at 502-1 through 502-N, 504; Specification: page 14 line 1 – page 15 line 15}**

a relationship manager for associating the plurality of devices when connected to the network with the plurality of service providers. **{Figure 5 at 510, 502-1 through 502-N, 504,**

506-1 through 506-M; Specification: page 14 line 1 – page 15 line 15}

1. A method of managing a relationship between a device and a service provider comprising:

{Figure 3 at 310, 330; Figure 5 at 502-1 through 502-N, 506-1 through 506-M;

Specification: page 10 line 21 – page 13 line 5; page 14 line 1 – page 15 line 15}

receiving at a service aggregator a first information from the device **{Figure 3 at 302, 320, 310; Figure 5 at 508, 502-1 through 502-N; Specification: page 10 line 21 – page 13 line 5; page 14 line 1 – page 15 line 15}**, the service aggregator having information about one or more service providers **{Figure 3 at 320, 330; Figure 5 at 508, 506-1 through 506-M; Specification: page 10 line 21 – page 13 line 5; page 14 line 1 – page 15 line 15}** and the service aggregator configured to select a specific service provider from the one or more service providers for the device based on the first information; and **{Figure 3; Figure 5; Specification: page 10 line 21 – page 13 line 5; page 14 line 1 – page 15 line 15}**

transmitting a second information from the service aggregator to the device **{Figure 3 at 320, 304, 310; Specification: page 10 line 21 – page 13 line 5}** directing the device to communicate with the specific service provider, **{Figure 3 at 310, 306, 308, 330; Specification: page 10 line 21 – page 13 line 5}** the second information being based on the first information received from the device, wherein the first information is sent automatically from the device to the service aggregator. **{Figure 3 at 304, 302, 310, 320; Figure 5; Specification: page 10 line 21 – page 13 line 5; page 14 line 1 – page 15 line 15}**

2. The method of claim 1 further comprising the service aggregator communicating information about the device to the specific service provider. **{Figure 3 at 320 to 330; Figure 5 at 508 to**

506-1 through 506-M; Specification: page 10 line 21 – page 13 line 5}

13. A method of managing a relationship between a device and a service provider, comprising:

{Figure 3 at 310, 330; Figure 5 at 502-1 through 502-N, 506-1 through 506-M;

Specification: page 10 line 21 – page 13 line 5; page 14 line 1 – page 15 line 15}

initially receiving at a service aggregator connected to a network first information from the device when the device is connected to the network, the service aggregator having information about one or more service providers and the service aggregator configured to select a specific service provider from the one or more service providers for the device based on the first information; **{Figure 3; Figure 4 at 402; Figure 5; Specification: page 10 line 21 – page 15 line 15}**

secondly transmitting a second information from the service aggregator to the device directing the device to communicate with the specific service provider, the second information being based on the first information received from the device, wherein the first information is sent automatically from the device to the service aggregator; **{Figure 3; Figure 4 at 402; Figure 5; Specification: page 10 line 21 – page 15 line 15}**

determining a new connection event from the device; and **{Figure 4 at 404;**

Specification: page 14 lines 6 – 23}

sending a new connection message to the device upon said determining said new connection event from said device; **{Figure 4 at 408; Specification: page 14 lines 6 – 23}**

determining and optionally updating the device upon said sending said new connection message to the device; **{Figure 4 at 410, 412; Specification: page 14 lines 6 – 23}**

sending messages to the device upon said determining and optionally updating the

device; **{Figure 4 at 424; Specification: page 14 lines 6 – 23}**

receiving user input from the device upon said sending messages to the device; and

{Figure 4 at 416; Specification: page 14 lines 6 – 23}

configuring the device upon said receiving user input from the device. **{Figure 4 at 418;**

Specification: page 14 lines 6 – 23}

//

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Issue: Whether Claims 27, 19, 1, 2, and 13 are patentable under 35 U.S.C. 102(e) in view of Anderson et al. (U.S. Pat. No. 6636259) (hereinafter Anderson).

//

VII. ARGUMENT

Issue: Whether Claims 27, 19, 1, 2, and 13 are patentable under 35 U.S.C. 102(e) in view of Anderson et al. (U.S. Pat. No. 6636259) (hereinafter Anderson).

FACTS PERTINENT TO THE ISSUE

The following enumerated Findings of Fact (FOF) are believed to be supported by a preponderance of the evidence.

File History

01. Applicant has requested several times (e.g. Response to OA, 17 March 2008, page 8) for a substantive examination of claims not specifically addressed. The Examiner's response (Office Action, 05/30/2008, page 10, lines 1-9) misquoted Applicant and additionally failed to address claims 19-24 and 27-28. The Examiner then turns around and complains that Applicant has failed to comply by arguing general allegations rather than specificity.

We find the Examiner's conduct not in accordance as contemplated by 37 CFR 1.104 because the Examiner has failed to indicate where each and every limitation of the claim may be found in the cited art. Applicant cannot respond with specificity if the Examiner has failed to indicate where each and every limitation of the claim may be found in the cited art.

Anderson

02. Anderson is directed to automatically configuring a web-enabled digital camera to access the Internet (Anderson Abstract). Anderson discloses an overall structure of cameras (e.g.

Fig. 1 at 12, 14, 28) which connect to a service provider (e.g. Fig. 1 at Service Provider A, etc.) which eventually connects to a Photo-Sharing Service (e.g. Fig. 1 at 16) through the Internet.

03. Anderson in Fig. 4A and 4B shows a flow chart of a preferred embodiment (Anderson, col 3, lines 3-6). Before a connection is established with an Internet Service Provider (ISP) (Fig. 4A at 118), the user must press "send" on the camera (Fig. 4A at 106, Anderson col 9, lines 66-67).

04. Anderson discloses a public network (Anderson at Abstract) and use of the Internet (Anderson at Title; Fig. 1, Fig. 4A and 4B, etc.). Official Notice is taken that the Internet is a public network and a network.

Facts Related to Claim Construction

05. The disclosure contains no explicit lexicographic definition of "aggregator" or "provider" but rather a functional definition of what each does in the specification as originally filed. Accordingly, an ordinary dictionary definition is guidance.

06. The ordinary and customary meaning of "aggregator" is that of bringing together. Therefore a "service aggregator" is one that brings together services.

07. The ordinary and customary meaning of "provider" is that of supplier. Therefore a "service provider" is a service supplier.

Facts Related To The Level Of Skill In The Art

08. Appellant has addressed the level of ordinary skill in the pertinent art.

Principles of Anticipation Law under 35 U.S.C. § 102

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference. *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631 (Fed. Cir. 1987). "When a claim covers several structures or compositions, either generically or as alternatives, the claim is deemed anticipated if any of the structures or compositions within the scope of the claim is known in the prior art." *Brown v. 3M*, 265 F.3d 1349, 1351 (Fed. Cir. 2001). "The identical invention must be shown in as complete detail as is contained in the . . . claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236 (Fed. Cir. 1989). The elements must be arranged as required by the claim, but this is not an *ipsissimis verbis* test, i.e., identity of terminology is not required. *In re Bond*, 910 F.2d 831, 832 (Fed. Cir. 1990).

Principles of Claim Construction

Claims must be "given their broadest reasonable interpretation consistent with the specification" (*Phillips v. AWH Corp.*, 415 F.3d 1303, (Fed. Cir. 2005)), "in light of the specification as it would be interpreted by one of ordinary skill in the art." (*In re Am. Acad. of Sci. Tech. Ctr.*, 367 F.3d 1359, (Fed. Cir. 2004)). Words of the claim must be given their plain meaning unless the plain meaning is inconsistent with the specification (*In re Zletz*, 893 F.2d 319, (Fed. Cir. 1989), *Chef America, Inc. v. Lamb-Weston, Inc.*, 358 F.3d 1371 (Fed. Cir. 2004)). Applicant may be their own lexicographer and claims are to be interpreted by that definition (*In re Paulsen*, 30 F.3d 1475 (Fed. Cir. 1994)). However, where a preamble sets forth a generic "method" with no field of endeavor set forth, or sets forth no machine or structure with which to further interpret or

limit the claimed invention, the claim is given its broadest reasonable interpretation in the English language for "a method" and looked to on how to interpret "a method" generically. *Ex parte Carlson*, BPAI 2006-003412, p5 ll12-22 (June 30, 2010).

//

Analysis

Claim 27 Rejection under 35 U.S.C. § 102(e) – Anderson

The Examiner argues at 27. in Office Action (Office) dated: 06/30/2009:

27. Claims 19 - 28 are rejected for similar reasons stated above.

Based on FOF 01, Applicant is unable to address issues not articulated, however Applicant can point with specificity to at least several limitations claimed which Anderson does not anticipate.

First, Applicant's claim 27 recites among other things "receiving information from a device automatically when the device is connected to a network". (Emphases added) Anderson does not anticipate this because Anderson requires a user to push "Send" on a camera before information is sent to a network (FOF 03, FOF 04.).

Second, Applicant's claim 27 recites among other things "determining if the device has access rights to a service provider after said receiving information from said device". (Emphases added.) Anderson does not anticipate this because Anderson connects to a service provider (FOF 02) without determining anything.

For the above reasons alone, Anderson does not anticipate what Applicant has claimed.

//

Claim 19 Rejection under 35 U.S.C. § 102(e) – Anderson

The Examiner argues at 27. in Office Action dated: 06/30/2009:

27. Claims 19 - 28 are rejected for similar reasons stated above.

Based on FOF 01, Applicant is unable to address issues not articulated, however applicant can point with specificity to at least several limitations claimed which Anderson does not anticipate.

First, Applicant's claim 19 recites among other things "a plurality of devices sending communications automatically with information about themselves when connected to the network". (Emphases added) Anderson does not anticipate this because Anderson requires a user to push "Send" on a camera before information is sent to a network (FOF 03, FOF 04.).

Second, Applicant's claim 19 recites among other things "a relationship manager for associating the plurality of devices when connected to the network with the plurality of service providers". (Emphases added) Nowhere does Anderson disclose a relationship manager, therefore Anderson does not anticipate Applicant's limitation of "a relationship manager for associating the plurality of devices when connected to the network with the plurality of service providers".

For the above reasons alone, Anderson does not anticipate what Applicant has claimed.

//

Claim 1 Rejection under 35 U.S.C. § 102(e) – Anderson

The Examiner at Office 4, 5., and 6. argues:

4. Referencing claim 1, as closely interpreted by the Examiner, Anderson teaches a method of managing a relationship between a device and a service provider comprising:
5. receiving at a service aggregator a first information from the device, the service aggregator having information about one or more service providers and the service aggregator configured to select a specific service provider from the one or more service providers for the device based on the first information, (e.g. col. 2, line 59 - col. 3, line 14 & col. 4, lines 9-37 & col. 9, line 25 - col. 10, lines 36, automatic and gateway server 18); and
6. transmitting a second information from the service aggregator to the device directing the device to communicate with the specific service provider, the second information being based on the first information received from the device, wherein the first information is sent automatically from the device to the service aggregator, (e.g. col. 2, line 59 - col. 3, line 14 & col. 4, lines 9 - 37 & col. 9, line 25 - col. 10, lines 36).

The Examiner at 5. first argues that Anderson discloses: receiving at a service aggregator a first information from the device, the service aggregator having information about one or more service providers and the service aggregator configured to select a specific service provider from the one or more service providers for the device based on the first information, (e.g. col. 2, line 59 - col. 3,

line 14 & col . 4, lines 9 - 37 & col. 9, line 25 - col. 10, lines 36, automatic and gateway server 18)

Firstly, Anderson fails to disclose a service aggregator and a service provider. The Examiner argues that a “gateway” can be considered an “aggregator” but provides no basis for this or support for this in Anderson. The Examiner again (as many times before) simply states a conclusion devoid of any reasoning or support. Applicant has called multiple times to contact the examiner in an attempt to try and understand why the Examiner considers an aggregator the same as a provider, the latest being December 29, 2009. The Examiner has refused to return any phone calls. As detailed with specificity in the application and in numerous office action responses, an aggregator and a provider are not the same. Applicant incorporates herein all five (5) prior office action responses and the supporting Exhibits therein. Per **Principles of Claim Construction** Applicant has clearly defined the distinction of a provider and an aggregator in the application as filed (e.g. Record), has provided support for this position with Exhibits (e.g. **EVIDENCE APPENDIX (C)(4)-(9)**) and still the Examiner provides nothing but conclusory statements. The latest being a “gateway” can be considered an “aggregator”. Applicant submits herewith, in the Anderson art arena of the Internet a definition of a “gateway” – see **EVIDENCE APPENDIX (C)(10)**. The definition is not new evidence but rather what a person having ordinary skill in the art already knows, that is the level of ordinary skill in the art. Also submitted is a dictionary definition in case the Board decides to use the dictionary definition, since Anderson fails to define “gateway” – see **EVIDENCE APPENDIX (C)(11)**. Additionally, submitted are two references from the era of Applicant’s filing, see **EVIDENCE APPENDIX (C)(12)** which shows a gateway server per Anderson terminology, and **EVIDENCE APPENDIX (C)(13)** also from the same era (note the latest edition simply added instructions

for installation from a CD, and therefore is considered from the 2000 date). Clearly seen from all these definitions and references, a “gateway” is not an aggregator.

Secondly, Applicant's claim specifies “receiving at a service aggregator a first information from the device” “wherein the first information is sent automatically from the device to the service aggregator”. Anderson does not anticipate this because Anderson requires a user to push “Send” on a camera before information is sent (FOF 03).

For the above reasons alone, Anderson does not anticipate what Applicant has claimed.

//

Claim 2 Rejection under 35 U.S.C. § 102(e) – Anderson

The Examiner at Office 7. argues:

7. Referencing claim 2, as closely interpreted by the Examiner, Anderson teaches comprising the service aggregator communicating information about the device to the specific service provider, (e.g. col. 10, lines 54-63).

As closely interpreted by the Applicant, Anderson fails to teach “comprising the service aggregator communicating information about the device to the specific service provider” as claimed by Applicant because taken as a whole Anderson does not teach what the Examiner incorrectly considers to be the aggregator the “gateway” communicating information about the device to the specific service provider after having received automatically a first communication from the device.

Additionally at the e.g. section cited by the Examiner, Anderson specifically states:

The camera receives the information packet in step 126, and writes a configuration file to memory 82a containing the ID, password, and default action list in step 128. The camera then displays the action list on the camera's LCD screen for selection by the user in step 130. The camera may optionally display the user's account information as well.

In an alternative preferred embodiment where security is a concern, the camera 14 first logs off the special ISP and the gateway accounts, and then reconnects using new ISP and gateway accounts in order to retrieve the action lists 48.

Nowhere is there a "gateway" to a new ISP communication. Rather as explained and shown by Anderson (e.g. Fig. 4B) there is only a "gateway" to camera communication (e.g. Fig. 4B at 124).

Additionally, the communication from the "gateway" to the camera does not teach "information about the device", rather as Anderson clearly shows the "information" sent is "The server then sends user account information to the device, including an account ID and password. (Abstract)", "...user account information to the digital camera, the user account including an account ID and an account password (claim 19.)" NOT information about the camera.

For the above reasons alone, Anderson does not anticipate what Applicant has claimed.

//

Claim 13 Rejection under 35 U.S.C. § 102(e) – Anderson

The Examiner at Office 13, argues:

13. initially receiving at a service aggregator connected to a network first information from the device when the device is connected to the network, the service aggregator having information about one or more service providers and the service aggregator configured to select a specific service provider from the one or

more service providers for the device based on the first information, (e.g. col. 2, line 59 - col. 3, line 14 & col. 4, lines 9 - 37 & col. 9, line 25 - col. 10, lines 36);

However Anderson taken as a whole and at the cited sections does not disclose "initially receiving at a service aggregator connected to a network first information from the device when the device is connected to the network". FOF 03.

Reference col. 2, line 59 - col. 3, line 14 discloses the Figures and establishing an account, not initially receiving at a service aggregator connected to a network first information from the device when the device is connected to the network.

Reference col. 4, lines 9 - 37 discloses entity ID information, and the gateway receiving this, not initially receiving at a service aggregator connected to a network first information from the device when the device is connected to the network.

Reference col. 9, line 25 - col. 10, lines 36 discloses Fig. 4A and 4B operation and as noted above (FOF 03) the user is required to push send, therefore it does not anticipate initially receiving at a service aggregator connected to a network first information from the device when the device is connected to the network.

The Examiner at Office 14. argues:

14. secondly transmitting a second information from the service aggregator to the device directing the device to communicate with the specific service provider, the second information being based on the first information received from the device, wherein the first information is sent automatically from the device to the service aggregator, (e.g. col. 2, line 59 - col. 3, line 14 & col. 4, lines 9 - 37 & col. 9, line 25 - col. 10, lines 36);

These cited references are a re-hash of the ones at 13. Anderson does not anticipate "wherein the first information is sent automatically from the device to the service aggregator" because first, per FOF 02, Anderson connects first to a service provider, and second, per FOF 03 the user is required to initiate transmission, and third, Anderson does not have a service aggregator that received the first communication from the device.

The Examiner at Office 15. argues:

15. determining a new connection event from the device, (e.g. col. 10, lines 38 - 53);

Firstly, Anderson fails to anticipate what Applicant has claimed because Anderson's sequence is not what Applicant has claimed. Applicant sends a second information from the service aggregator to the device directing the device to communicate with the specific service provider, whereas Anderson is communicating with the same Photo-Sharing Service 16.

Secondly, Anderson sets up an account on a first-time connection whereas Applicant sends a new connect message.

The Examiner at Office 16. argues:

16. sending a new connection message to the device upon said determining said new connection event from said device, (e.g. col. 10, lines 54 - 64);

Anderson only sends a message to the camera after setting up a new account (col. 10 lines 37-43, Fig. 4B at 122 and then 124), which is not what Applicant claims.

The Examiner at Office 17. argues:

17. determining and optionally updating the device upon said sending said new connection message to the device, (e.g. col. 10, lines 30 - 37);

Anderson at col. 10, lines 30 - 37 discloses no updating. First, Anderson only "updates" an action list 48 (see Fig. 5) that is "... automatically downloaded to the user's camera 14 during a connection with the gateway server 18 and stored on the camera 14." (col. 8 lines 13-15) however these updates lists are done (see Fig. 4B at 124) before any new connection message is sent to the device and

second, the updated action list is not based on the new connection message to the device but rather what is stored in (see Fig. 2) the Photo-Sharing Service 16, Database 20, at a User Account 40, Action Lists 48. Thus, Anderson does not anticipate determining and optionally updating the device upon said sending said new connection message to the device.

The Examiner at Office 18. argues:

18. sending messages to the device upon said determining and optionally updating the device, (e.g. col. 10, lines 38 - 53);

Anderson at the cited section discloses steps 122 and 124 in Fig. 4B. As can be seen in Fig. 4B after the "updated" action list in step 124 is sent (which is received at step 126 by the camera (col. 10, line 54)) there is no further sending of a message to the camera (128 camera writes info to itself, displays action list 130, uploads images 132, displays progress bar 134, and disconnects 136). Thus Anderson fails to anticipate sending messages to the device upon said determining and optionally updating the device.

The Examiner at Office 19. argues:

19. receiving user input from the device upon said sending messages to the device, (e.g. col. 10, lines 15 - 29); and

Firstly, as explained above at 18. Anderson does not send messages to the device after updating the action lists, so it logically cannot then receive user input after such. Rather Anderson receives Upload images (Fig. 4 at 132) without sending a message after updating the action list (Fig. 4B at 124).

Secondly, the cited section deals with the camera (Fig. 4B at 116) checking to see if it has connected before not a new connect message from a service aggregator.

The Examiner at Office 20. argues:

20. configuring the device upon said receiving user input from the device, (e.g. col. 10, lines 38-53).

As noted at 19. Anderson does not send messages to the device after updating the action lists, so it logically cannot then receive user input after such. Additionally, Anderson at the cited section discloses steps 122 and 124 in Fig. 4B. As can be seen in Fig. 4B after the “updated” action list in step 124 is sent (which is received at step 126 by the camera (col. 10, line 54)) there is no configuring the camera, because there is no communication from the Photo-Sharing Service to the camera (the Photo-Sharing Service only receives the Uploaded images (Fig. 4B at 132). Thus Anderson fails to anticipate configuring the device upon said receiving user input from the device.

Finally, a simple comparison of Applicant's Figure 4 with Anderson's Fig. 4B clearly shows Anderson does not anticipate what Applicant has claimed.

//

CONCLUSION

Applicant submits that the rejection of dependent claims not specifically addressed, are addressed by Applicant's arguments to the independent claims on which they depend.

Applicant respectfully submits that the appealed claims in this application are patentable, and requests that the Board of Patent Appeals and Interferences direct allowance of all claims.

Respectfully submitted,

Heimlich Law

08/03/2010

/Alan Heimlich/

Date

Alan Heimlich / Reg 48808

Customer No. 40418

Attorney for Applicant(s)

5952 Dial Way
San Jose, CA 95129

Tel: 408 253-3860
Eml: alanheimlich@heimlichlaw.com

VIII. CLAIMS APPENDIX

27. A method of relationship management comprising:

receiving information from a device automatically when the device is connected to a network;

determining if the device has access rights to a service provider after said receiving information from said device; and

if the device has access rights to the service provider, sending information to the device on how to contact the service provider.

19. A system comprising:

a plurality of service providers in communication with a network;

a plurality of devices sending communications automatically with information about themselves when connected to the network; and

a relationship manager for associating the plurality of devices when connected to the network with the plurality of service providers.

1. A method of managing a relationship between a device and a service provider comprising:

receiving at a service aggregator a first information from the device, the service aggregator having information about one or more service providers and the service aggregator configured to select a specific service provider from the one or more service providers for the device based on the first information; and

transmitting a second information from the service aggregator to the device directing the

device to communicate with the specific service provider, the second information being based on the first information received from the device, wherein the first information is sent automatically from the device to the service aggregator.

2. The method of claim 1 further comprising the service aggregator communicating information about the device to the specific service provider.

13. A method of managing a relationship between a device and a service provider, comprising:

initially receiving at a service aggregator connected to a network first information from the device when the device is connected to the network, the service aggregator having information about one or more service providers and the service aggregator configured to select a specific service provider from the one or more service providers for the device based on the first information;

secondly transmitting a second information from the service aggregator to the device directing the device to communicate with the specific service provider, the second information being based on the first information received from the device, wherein the first information is sent automatically from the device to the service aggregator;

determining a new connection event from the device; and

sending a new connection message to the device upon said determining said new connection event from said device;

determining and optionally updating the device upon said sending said new connection message to the device;

sending messages to the device upon said determining and optionally updating the

device;

receiving user input from the device upon said sending messages to the device; and
configuring the device upon said receiving user input from the device.

//

IX. EVIDENCE APPENDIX

Grouping of any Evidence for Claim purposes is for the convenience of reduced duplication and is NOT to be interpreted as the Grouping of Claims for Arguments under 37 C.F.R. § 41.37.

(A) Evidence for Claims on Appeal – Relied Upon

The following items listed below are hereby entered as evidence relied upon by the Examiner as to grounds of rejection for claims 1-25 to be reviewed on appeal. Also listed for each item is where said evidence was entered into the record by the Examiner.

(1) Copy US Patent 6636259 10-2003 (Anderson). This evidence was entered into the record by the Examiner at 3. of the Office Action dated 06/30/2009.

Copies of all References follows.

//



US006636259B1

(12) **United States Patent**
Anderson et al.

(10) **Patent No.:** **US 6,636,259 B1**
(45) Date of Patent: **Oct. 21, 2003**

(54) **AUTOMATICALLY CONFIGURING A WEB-ENABLED DIGITAL CAMERA TO ACCESS THE INTERNET**

6,167,469 A * 12/2000 Safai et al. 710/62
 6,226,752 B1 * 5/2001 Gupta et al. 713/201
 6,269,481 B1 * 7/2001 Perlman et al. 717/11
 6,502,195 B1 * 12/2002 Colvin 713/202

(75) Inventors: **Eric C. Anderson**, San Jose, CA (US);
Robert Paul Morris, Raleigh, NC (US)

* cited by examiner

(73) Assignee: **IPAC Acquisition Subsidiary I, LLC**,
 Peterborough, NH (US)

Primary Examiner—Wendy R. Garber
Assistant Examiner—Jacqueline Wilson

(74) *Attorney, Agent, or Firm*—Sawyer Law Group LLP

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 32 days.

(57) **ABSTRACT**

(21) Appl. No.: **09/625,824**

(22) Filed: **Jul. 26, 2000**

(51) **Int. Cl.**⁷ **H04N 5/232**

(52) **U.S. Cl.** **348/211.3; 348/211.12; 348/14.04**

(58) **Field of Search** **348/14.01, 14.02, 348/14.04, 14.08, 14.09, 552, 143, 156, 157, 211.3, 211.12; 709/322**

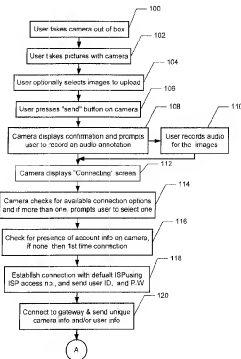
(56) **References Cited**

U.S. PATENT DOCUMENTS

5,430,827 A * 7/1995 Rissanen 704/272
 5,737,491 A * 4/1998 Allen et al. 704/270
 5,905,736 A * 5/1999 Ronen et al. 370/546
 6,064,671 A * 5/2000 Killian 370/389
 6,067,571 A * 5/2000 Igarashi et al. 709/232

The present invention provides a method and system for automatically establishing a user account and for configuring a hand-held electronic device for accessing a site on a public network using the user account. The method and system include establishing a connection to a website server, which is responsible for establishing and maintaining website accounts. The presence of account information on the electronic device camera is then checked, and if none are found, it is determined that this is a first-time connection and information uniquely identifying the electronic device is sent to the website server so that the server can set-up the account information based on the electronic device information. The server then sends user account information to the device, including an account ID and password. The user account information is then stored on electronic device for use the next time the electronic device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network.

22 Claims, 6 Drawing Sheets



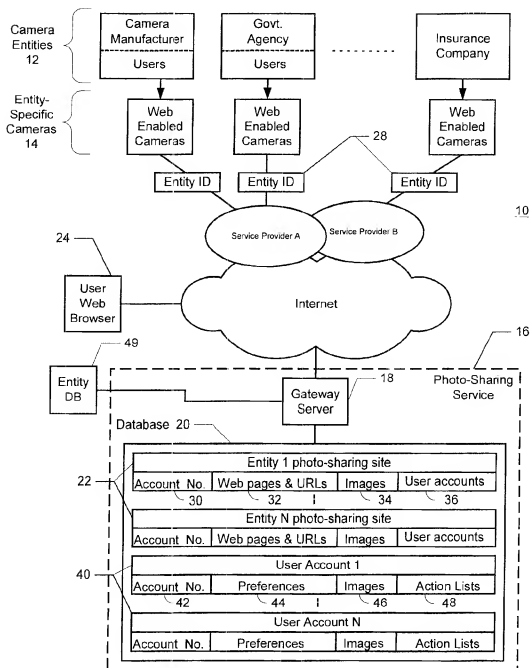


FIG. 1

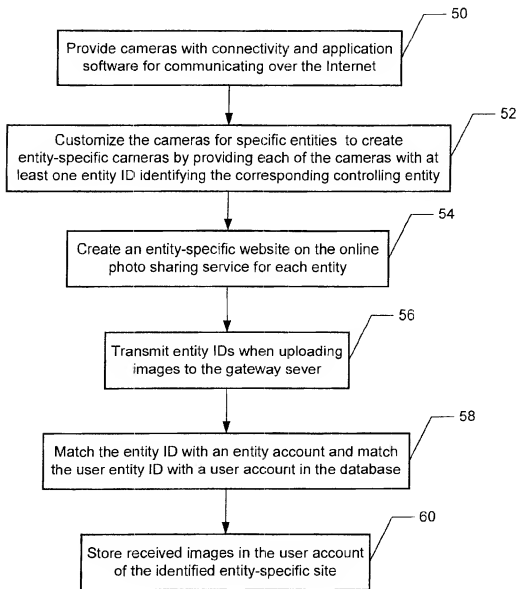


FIG. 2

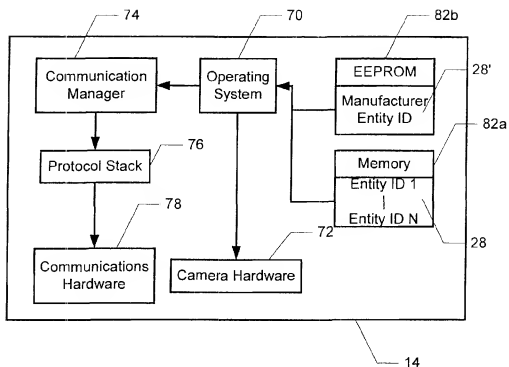


FIG. 3

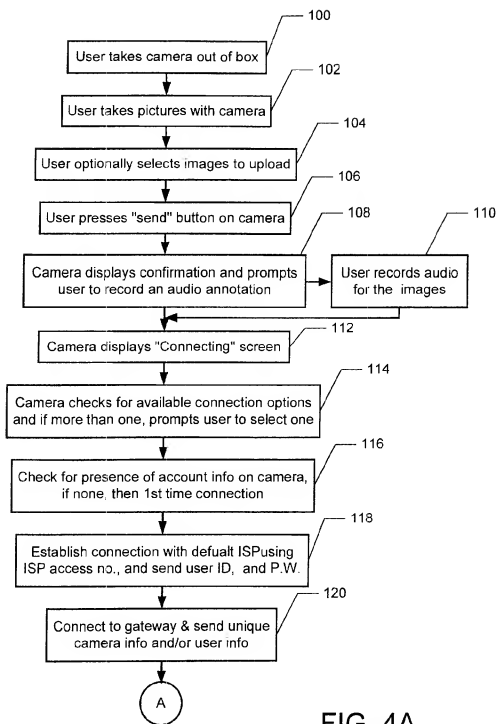


FIG. 4A

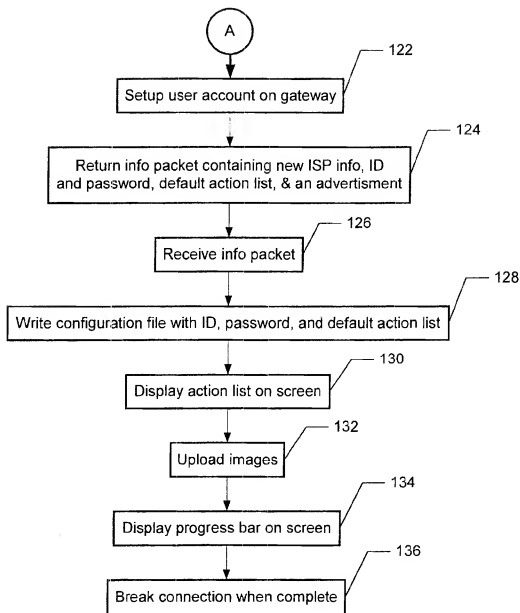


FIG. 4B

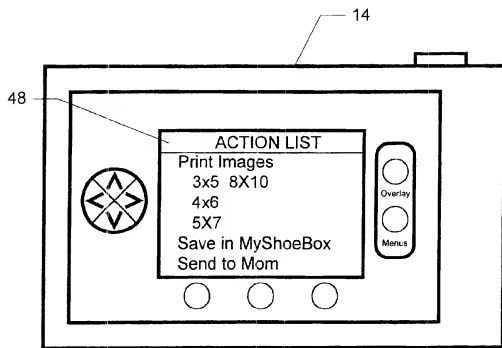


FIG. 5

1

AUTOMATICALLY CONFIGURING A WEB-ENABLED DIGITAL CAMERA TO ACCESS THE INTERNET

CROSS-REFERENCE TO RELATED APPLICATIONS

The present invention is related to co-pending U.S. patent application Ser. No. 09/625,398 entitled "Method and System For Hosting Entity-Specific Photo-Sharing Websites For Entity-Specific Digital Cameras"; and to co-pending U.S. patent application Ser. No. 09/626,418, entitled "Method And System For Selecting Actions To Be Taken By A Server When Uploading Images," which are assigned to the assignee of the present application and filed on the same date as the present application.

FIELD OF THE INVENTION

The present invention relates to a method and system for customizing digital cameras to upload images to an entity-specific photo-sharing websites, and more particularly to automatically configuring a web-enabled digital camera to access the Internet.

BACKGROUND

As the popularity of digital cameras grows, the desire of digital camera users to share their images with others will also continue to grow. New digital camera owners typically try to share their images based on the paradigm of film cameras, in which images are printed on paper and then placed into a photo album. The most straightforward approach to do this with a digital camera is to connect the digital camera directly to a printer to create the prints, and then manually insert the images into a photo album. Users often find this process somewhat complicated and restrictive because standard printers can only print images in limited sizes and requires particular types of paper. And even after the photo album has been assembled, the printed images are not easily shared with many people.

The best approaches to photo-sharing take advantage of the Internet. One such approach is for users to store the digital images on a PC and then send the images to others using email. Several Internet companies now offer an even more convenient approach by providing photo-sharing websites that allow users to store their images for free and to arrange the images into web-based photo albums. Once posted on a photo-sharing website, others may view the images over the Internet.

While convenient for storing digital images, getting the images to the photo-sharing websites can be challenging for users. Most commonly, users must upload their images from the digital camera to a PC using a cable or IrDA, or by inserting the camera's flash card into the PC. From the PC, the user logs onto the Internet and uploads the images to a photo-sharing website. After uploading the images, the user works on the website to arrange the images into web albums and to add any textual information.

Although today's approach for storing images from a digital camera onto a web photo-sharing website and for creating web photo albums works reasonably well, two problems remain that hinder the mainstream adoption of web-based photo-sharing. One problem is that this approach requires the use of a PC, notebook computer, or PDA. While many digital camera users today have PC's, most of those users are early adopters of technology. There are many other consumers who would purchase a digital camera, but are

2

reluctant to do so because they do not yet own a PC or are intimidated by them.

In an effort to address this problem, the assignee of the present application developed an approach to uploading images to the web that doesn't require the use of a PC. In this approach, an email software application is loaded into a digital camera capable of running software that allows the user to e-mail the images directly from the camera. The user simply connects his or her digital camera to a cellphone or modem, runs the e-mail application, and selects the desired images and the email recipients. The selected images are then sent to the recipients as e-mail attachments.

Although emailing photos directly from the camera allows users who do not own a PC to share images over the Internet, these users must still establish accounts with both an Internet service provider (ISP) and the photo-sharing website before being able to post their images. These accounts must also be set-up by PC users as well. For techno savvy users who use a PC to upload the images to the photo-sharing website, establishing the accounts may not be a bother, but even these users may not always have their PC's handy, such as when on vacation, for instance. And for non-PC users, establishing the accounts by entering account information on the digital camera itself may prove to be a cumbersome, if not a daunting, task.

Accordingly, what is needed is an improved method for uploading images from a digital camera to a photo-sharing website on the Internet. In order for online photo-sharing to become more mainstream, an approach that doesn't require a PC or PC expertise and that reduces complexity for the user is required. The present invention addresses such a need.

SUMMARY OF THE INVENTION

The present invention provides a method and system for automatically establishing a user account and for configuring a hand-held electronic device for accessing a site on a public network using the user account. The method and system include establishing a connection to a website server, which is responsible for establishing and maintaining website accounts. The presence of account information on the electronic device camera is then checked, and if none are found, it is determined that this is a first-time connection and information uniquely identifying the electronic device is sent to the website server so that the server can set-up the account information based on the electronic device information. The server then sends user account information to the device, including an account ID and password. The user account information is then stored on electronic device for use the next time the electronic device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network.

According to the method and system disclosed herein, a user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network with the electronic device.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG 1 is a block diagram illustrating an online photo-sharing system in accordance with a preferred embodiment of the present invention.

FIG 2 is a flow chart illustrating a business method for hosting entity-specific photo-sharing websites for entity-specific image capture devices in accordance with a preferred embodiment of the present invention.

3

FIG. 3 is a diagram showing a preferred embodiment of the connectivity and application software of the camera.

FIGS. 4A and 4B illustrate a flow chart of a process for automatically configuring a web-enabled digital camera to access the Internet in a preferred embodiment of the present invention.

FIG. 5 is a block diagram illustrating an example action list 48 displayed on the LCD screen of the camera.

DETAILED DESCRIPTION

The present invention relates to an automatic system for uploading images from a digital camera to entity-specific photo-sharing websites and for automatically establishing accounts. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

FIG. 1 is a block diagram illustrating an online photo-sharing system 10 in accordance with a preferred embodiment of the present invention. The system includes a plurality of camera controlling entities 12 that produce, own, or otherwise control a set of digital cameras 14, and an online photo-sharing service 16. The online photo-sharing service 16 includes a gateway server 18 and an entity/account database 20. The various camera controlling entities 12 contract with the photo-sharing service 16 to transparently host customized photo-sharing websites 22 for each entity, which are referred to herein as entity-specific photo-sharing websites 22. The entity-specific photo-sharing websites 22 are each accessible to the user through the entity's existing Internet site (not shown), and thus appear to users as though the entity-specific photo-sharing websites 22 are hosted by the corresponding entity. According to a preferred embodiment of the present invention, the cameras 14 for a particular entity are customized for that entity to create entity-specific cameras 14, such that when the cameras 14 connect to the Internet, the cameras 14 automatically upload their images to the photo-sharing website of the corresponding entity. In a further aspect of the present invention, the photo-sharing service 16 automatically stores the images in a web album, which is viewable over the Internet by a user's web browser 24.

As used herein, a camera controlling entity 12 is any entity that makes, owns, sells, or controls digital cameras 14, and therefore includes, camera manufacturers, companies, retailers, and end-users. One or more combination of these entities 12 may either contract with the photo-sharing service 16 to provide entity-specific websites 22 for their cameras 14, or have entity information transmitted to the photo-sharing service 16 from the cameras 14. Therefore, a camera controlling entity 12 may include a single entity 12 or a hierarchical relationship of entities 12.

An example of a single entity 12 includes an insurance company that contracts with the photo-sharing service 16 to have all digital cameras 14 used by their agents to transmit their images to a customized insurance photo-sharing website. Examples of a hierarchical relationships of entities 12 includes a camera manufacturer, such as Nikon, that contracts with the photo-sharing service 16 to have all Nikon digital cameras 14 transmit their images to the customized

4

Nikon photo-sharing website. Since the images of different users must be distinguished, each user of a Nikon camera 14 would also constitute an entity within the Nikon website so that the images from different users can be distinguished. Other examples of hierarchical entity relationships include a retailer and its consumers, a real estate agency and its agents, community groups and its members, and government agencies and its employees, for instance.

In a preferred embodiment, the cameras 14 are customized for their respective entities 12 by providing the cameras 14 with software for transmitting entity ID information 28 identifying its controlling entity 12 to the photo-sharing service 16. The photo-sharing service 16 in conjunction with the gateway server 18 and the entity/account database 20 hosts the entity-specific photo-sharing websites 22. Each entity-specific website 22 is identified in the database 20 by an entity account number 30 and includes the web pages and URIs 32 comprising the website, the images and web albums 34 stored on the website, and the user account numbers 36 of authorized users. The database 20 also includes user accounts 40, each of which comprises a user account number 42, user preferences 44, the user's images 46, and action lists 48, explained further below.

The gateway server 18, which communicates with the cameras 14 during image uploading, receives one or more entity IDs 28 from each camera 14 and matches the entity ID 28 with an entity account 30 in the database 20. The images are then automatically associated with the photo-sharing website 22 of the identified entity 12 and/or the identified user.

After the images are uploaded, a user of the camera 14 may visit the online photo-sharing website 22 over the Internet to view the images via a web browser 24. Since the photo-sharing websites 22 are transparently hosted by the photo-sharing service 16, each photo-sharing website 22 appears as though it is hosted by the entity itself, rather than the third party service.

In one embodiment, the cameras 14 may connect to the Internet via a service provider 26, which may include a wireless carrier and/or an Internet service provider (ISP) that is capable of servicing many devices simultaneously. In a preferred embodiment, each of the cameras 14 is provided with wireless connectivity for connecting to the Internet, and are therefore so called "web-enabled" devices, although a wired connection method may also be used.

The cameras 14 may be provided with wireless connectivity using anyone of a variety of methods. For example, a cellphone may be used to provide the digital camera 14 with wireless capability, where the camera 14 is connected to the cellphone via a cable or some short-range wireless communication, such as Bluetooth. Alternatively, the camera 14 could be provided with built-in cellphone-like wireless communication. In an alternative embodiment, the digital camera 14 is not wireless, but instead uses a modem for Internet connectivity. The modem could be external or internal. If external, the camera 14 could be coupled to modem via any of several communications means (e.g., USB, IEEE1394, infrared link, etc.). An internal modem could be implemented directly within the electronics of camera 14 (e.g., via a modem ASIC), or alternatively, as a software-only modem executing on a processor within camera. As such, it should be appreciated that, at the hardware connectivity level, the Internet connection can take several forms. Hence, it should be appreciated that the present invention is not limited to any particular method of accessing the Internet.

5

In a preferred embodiment, the entity-specific websites 22 are customized to seamlessly integrate into the entity's existing website by following the look and feel of the entity's existing website. The entity-specific websites 22 are hosted on the photo-sharing service 16, but a link to the entity-specific websites 22 may be provided on the homepage of the corresponding entity's existing website. Thus, in order to view a web album on an entity-specific website 22, the user must visit the entity's existing website and click the link to the entity-specific website 22, where the user's browser 24 will be transparently directed to the photo-sharing service 16 and be provided with the web pages 32 of the entity-specific website 22.

As an example of the operation of the photo sharing system 10, consider the following scenario. Assume that Minolta and Nikon are entities 12 that have contracted with the photo-sharing service 16, and that the photo-sharing service 16 hosts a photo-sharing website 22 for Minolta and a photo-sharing website 22 Nikon. The Minolta cameras 14 would be provided the entity ID 28 for Minolta and the Nikon cameras 14 would be provided the entity ID 28 for Nikon. When the Minolta and the Nikon cameras 14 send sets of images to the photo-sharing service 16, the gateway server 18 would distinguish the cameras 14 by the entity IDs 28 and would direct the set of images received from Minolta cameras 14 to Minolta's photo-sharing website, and would direct the images from Nikon cameras 14 to Nikon's photo-sharing website. To view the images, the owners of the cameras 14 would use a browser 24 on their PC or PDA to visit the URL of the Minolta or Nikon photo-sharing websites 22. In one preferred embodiment, the photo-sharing service 16 sends the URL of the entity-specific website 22 directly to the camera 14 for display to inform the user of the address.

According to the present invention, the photo-sharing service 16 provides business-to-business and business-to-consumer business models. The service is business-to-business because the service provides companies, such as camera manufacturers, with a complete end-to-end solution for their cameras 14. The solution includes customized software for their cameras 14 for sending images over the internet, and an internet website for storing images from those cameras 14 on a branded website that appears to be hosted by the company. The service is business-to-consumer because it allows users of digital cameras 14 with an automatic solution for uploading captured images from a digital camera 14 to an online photo-sharing website, without a PC.

According to one preferred embodiment of the present invention, the photo-sharing service 16 provides a method of doing business whereby the photo-sharing service 16 shares revenue based on the hierarchical relationship of the entities 12. For example, if the photo sharing service 16 charges a fee for receiving and/or storing the images received from the entity-specific cameras 14, then the photo sharing service 16 may share a portion of the fee with the manufacturer and/or third party supplier of the camera 14 that uploaded the images, for instance. Revenue may also be shared with the wireless service provider providing the connection with the photo sharing service 16.

FIG. 2 is a flow chart illustrating a business method for hosting entity-specific photo-sharing websites for entity-specific image capture devices, such as digital cameras, in accordance with a preferred embodiment of the present invention. First, the cameras 14 are provided with connectivity and application software for communicating over the Internet in step 50. In a preferred embodiment, this step is

6

performed during camera 14 manufacturing to provide off-the-shelf web enabled cameras 14. The cameras 14 are also customized for specific entities 12 to create entity-specific cameras 14 by providing each of the cameras 14 with at least one entity ID 28 identifying the corresponding controlling entity in step 52.

Referring now to FIG. 3, a diagram showing the preferred embodiment of the connectivity and application software of the camera 14 and the entity ID 28 information is shown. Preferably, the camera 14 includes a microprocessor-based architecture that runs an operating system 70 for controlling camera hardware 72 and overall functionality of the camera 14 (e.g., taking pictures, storing pictures, and the like). An example of such an operating system 70 is the Digita™ Operating Environment developed by the assignee of the present application. The camera 14 also includes communication manager 74 software, and a TCP/IP protocol stack 76, that enables communication via the internet, as is well-known in the art. The entity ID information 28 and captured images may be stored in one or more types of memories 82.

For hierarchical entity relationships, the cameras 14 are provided with hierarchical entity IDs 28; one entity ID 28 identifying the entity, and a second entity ID 28 identifying the end-user. Whether there are one or more entity IDs 28, the entity ID 28 of the camera manufacturer may always be provided. Camera 14 customization may occur either during manufacture or anytime thereafter. In a preferred embodiment, the manufacturer entity ID 28 is provided at the time of manufacturing and is stored in an EEPROM 82b, while the entity IDs 28 for other entities 12, such as companies and end-users, are loaded into the camera 14 subsequent to manufacturing and are stored in flash memory 82a or the EEPROM 82b.

Customization that occurs subsequent to manufacture may be implemented using several methods. The first method is to manufacture the cameras 14 with an application programming interface (not shown) for accepting a subsequently loaded software application that specifies the entity ID's 28. The application may come preloaded on a flash card, which is then inserted into the camera 14 by the user and stored in flash memory 82a. The application may also be wirelessly beamed into the camera. When executed in the camera, the software application transmits the appropriate entity IDs 28 to the gateway server 18.

The second method is to load a small file in the camera 14 specifying the entity IDs 28 from a removable memory or from a PC, and storing the file in a system folder within the camera's flash memory 82a. The camera 14 then accesses the file when an Internet connection is established. In a preferred embodiment, the communication manager 74 automatically extracts the manufacturing ID 28 and the entity ID 28 and transmits them to the gateway server 18. In this embodiment, the entity ID 28 is also stored in the EEPROM 82b and is factory set to zero (empty). Thus, unless the entity ID 28 is set, the manufacturing ID 28 may default as the highest controlling entity.

If, for example, a third party developer X contracts to provide custom camera software for camera manufacturer Z, then a custom entity ID will be issued for developer X and developer X will place the custom entity ID into the EEPROM 82b. Developer X is now a controlling entity 12, and may specify to the photo-sharing service 16 that a developer X entity-specific photo-sharing site 22 or developer X's own website be the destination for the uploaded images.

7

The protocol stack 76, under direction of the communications manager interfaces with the communications hardware 78 of camera. The protocol stack 76 includes software APIs and protocol libraries that interface with the communication manager 74, and communication hardware interface drivers that interfaces directly with the various communication hardware 72 the camera 14 must function with (e.g., a Bluetooth transceiver, etc.). The communication manager 74 communicates with operating system 70 and the IP protocol stack 76 to establish an Internet connection ant to transmit the entity ID 28 information and images from the memories 82a and 82b to the photo-sharing service 16.

In an alternative embodiment, rather than loading entity ID's 28 into the camera, a combination of the camera's serial number and the make and model number of the camera may be used as the entity ID 28. Entity specific cameras 14 may then be distinguished by providing a mapping of the camera serial numbers and product IDs to specific entities 12 in the database 20.

Although the camera 14 has been described in terms of a software-based customization solution, those with ordinary skill in the art will readily recognize that the camera 14 may also be provided with a hardware-based solution.

Referring again to FIG. 2, before or after camera customization, the entity-specific websites 22 are created for each entity contracting with the photo-sharing service 16 in step 54. Customization requires storage of entity information in the entity/account database 20 and creating and storing web page elements comprising the entity-specific photo-sharing website in the database 20. The entity-specific information stored in the database 20 may also include service levels, and enabled features for the entity-specific website 22. Features are components or services that may be provided on websites by the photo-sharing service 16, such as search functions, and online printing, for instance, but may be selectable by each entity for its own website. As an example, company X may provide customized cameras 14 for its employees, but may not wish to allow employees to print images from the company X photo website for security reasons. If so desired, company X may have the photo service disable this feature from their particular website.

In a preferred embodiment, the entity-specific websites 22 are not created from scratch, but are created by modifying a preexisting template. The template may include several different sections, such as A, B, C and D, for instance. Assuming for example that the template used to create a website for Nikon, and section A is used to specify the name of the entity then the name Nikon would be inserted into that section. Other entity-specific content would be used to fill out the remaining sections. The Web pages comprising the Nikon specific photo-sharing website would then be provided with URL's unique to that website. The entity's regular website would be modified to include a link to the entity's photo-sharing website 22. In addition, the entity-specific photo-sharing website would include a link back to the entity's website. Entities 12 may have entity photo-sharing websites 22 created for them in one of two ways; automatically by logging into the photo-sharing service 16 and manually customizing the templates, or by having the entity photo-sharing website created for them.

Referring still to FIG. 2, when a camera 14 establishes an internet connection with the gateway server 18, the camera 14 transmits its entity IDs 28 and/or user entity ID 28 when uploading user selected images to the gateway server 18 in step 56. In response, the gateway server 18 matches the entity ID 28 with an entity account in the database 20 and

8

matches the user entity ID 28 with a user account 40 in the database 20 in step 58. The images received are then stored in the user account 40 of the identified entity-specific website 22 in step 60.

Referring again to FIG. 1, each user account 40 in the database 20 may also include one or more action lists 48. According to the present invention, an action list 48 includes one or more items representing actions that the gateway server 18 should take with respect to uploaded images, such as where to store and/or send the images from a particular user or camera, for instance. As explained further below, the action list 48 stored on the database 20 under a user's account 40 are automatically downloaded to the user's camera 14 during a connection with the gateway server 18 and stored on the camera 14. When the user initiates an image upload, the action list 48 is displayed to the user so the user may easily select what actions the gateway server 18 should take with respect to the images by selecting the displayed action list items.

Examples of action list items include specifying that the uploaded images should be stored on the entity-specific photo website, sending the images to a list of email addresses, or even performing some type of analysis or calculation on the image data, for instance.

In a further aspect of the present invention, an action list item is not limited to, instructing the gateway server 18 to perform actions only within the photo-sharing service 16. Rather, an item in the action list 48 may also instruct the gateway server 18 to perform actions outside of the photo-sharing service 16, such as storing the images in an external database 49 of the entity 12. For instance, in the example where the entity 12 is a company, some users of the company's cameras 14 could have action lists 48 instructing the gateway server 18 to store uploaded images to the company's database, rather than to the company's photo-sharing site 22. Based on the action lists 48 and customization, the gateway server 18 may be programmed to automatically perform predefined tasks, such as creating new web albums, or a new page within an existing album, parse the images to extract sound files or other metadata, print images and mail them to designated addresses, and so on.

In a preferred embodiment, the action lists 48 may be created via several methods. In one method, the action list is created by the photo-sharing service 16 the first time the user's camera establishes a connection. That is, a default action list 48 is automatically created based on the entity ID when a user account 40 is first created. In a hierarchical entity relationship where the entity 12 is a company, a default action list 48 may be created to implement a workflow specified by the entity 12. In a hierarchical relationship where the entity 12 is camera manufacturer, for instance, a default action list 48 may be created instructing the gateway server 18 to store the user's images in a simulated "shoebox" on the entity-specific photo-sharing site 20. The user may then go online and create albums from the images in the shoebox as desired.

Another method is for the user to create the action list 48 online on the entity-specific photo-sharing site 20. The action list 48 may be created manually on the website 20 by the user navigating to the site 20 using a web browser 24, accessing her account, and manually creating the action list 48 or editing the action list 48 on the entity-specific site 22. The action list 48 may also be created automatically on the website 20 in response to user actions performed on the website, such as printing images, or creating a web album.

9

Alternatively, after performing an action, the user may be prompted whether they would like this action added to his or her action list 48. If so, the user clicks a check-box and the item is added the action list 48. In a preferred embodiment, any action list 48 created and edited on the photo-sharing site 20 are downloaded to the camera every time the camera 14 connects to the photo-sharing service 16 and made available for user selection on the camera 14 during the next upload.

Yet another method for creating an action list 48 is to allow the user to create the action list on the camera 14. The user may manually create an action list 48 by "typing" in predefined items on the camera. The user may also type in an email address as an action list item whereby when that item is selected, the uploaded images are stored as a web album on the entity-specific photo-sharing website 22 and the server 18 sends a notification to the specified recipient containing the URL to the web album page.

A method and system for hosting web-based photo-sharing websites and for customizing digital cameras to upload images to the entity-specific photo-sharing websites has been disclosed. According to the present invention, users of the customized cameras 14 can upload images to the Internet for storage and web photo album creation without the use of a PC.

In one embodiment described above, the present invention assumes that an ISP account has been established with the digital camera's service provider, and that users of cameras 14 belonging to a certain entity 12 may use the cameras 14 to upload images to the website of the entity 12. However, two problems with account setup remain. One problem is that just as with a PC and PDA, the user must first establish an ISP account before the camera 14 can establish Internet communication. The second account problem is that most websites, including the photo sharing sites 22, may require each user to establish a unique account before using the site to distinguish one user from another. Before being able to connect the web-enabled camera 14 to the Internet, the user must establish these two accounts by either entering account setup information on a PC or entering account setup information on the camera 14. Neither alternative is a convenient alternative for people who do not have the time nor inclination to do so.

In a further aspect of the present invention, the cameras and the photo sharing site are provided with software for automatically creating Internet and photo-sharing website accounts for each camera 14 upon first use, without requiring the user to first enter account information on a PC or on the camera 14.

Referring now to FIGS. 4A and 4B, a flow chart illustrating a process for automatically configuring a web-enabled digital camera to access the Internet in a preferred embodiment of the present invention. Although the process will be described in terms of automatically establishing Internet accounts for a digital camera without requiring the user to enter information, those with ordinary skill in the art will readily recognize that the present invention may be used to automatically establish Internet accounts for any type of portable electronic device.

The process assumes that a user has just acquired a digital camera 14 customized as described above, and has just taken the camera 14 out of its box in step 100. After taking pictures with the camera in step 102, the user may review the images in the camera's LCD screen and optionally select a set of images to upload to the photo sharing service 16 in step 104. The user then presses a "send" button on the camera in step 106 to upload the images.

10

In response, the camera displays a confirmation dialog screen on the camera and prompts the user to record an audio annotation for the images or to continue in step 108. The user may then choose to record audio for the images in step 110. After choosing to continue, or after recording audio, the camera displays a "connecting" dialog screen in step 112 to indicate to the user that the camera is establishing an Internet connection. At the same time, the camera checks for available connection options in step 114, and if more than one is found, the camera prompts the user to select one of the connection options. For example, the camera may be within range of a Bluetooth-equipped printer and a cellphone, so the user will be prompted to choose which device the camera should establish communication with.

The camera then checks for the presence of account information on the camera in step 116, and if there are none, the camera assumes that this is a first-time connection. According to the present invention, in order to allow the camera to make a first-time Internet connection, the camera is provided with default Internet service provider (ISP) information during manufacturing, including an ISP access number, and user ID and password (if required). The camera establishes connection with the default ISP in step 118 by dialing the preloaded access number, and by sending the preloaded user ID and password to the ISP. This special account may be configured so that the camera can only connect to the gateway server 18 (no other IP addresses may be allowed).

After connecting with the ISP, the camera connects to the gateway server 18 and sends unique camera information and/or user information in step 120. In a preferred embodiment of the present invention, a combination of the camera's serial number and the make and model number of the camera may be sent as the unique camera information. In another preferred embodiment, the user's e-mail address may be sent as the unique camera or user information.

Continuing with FIG. 4B, the gateway server 18 uses the unique camera information to set up a user account 40 in step 122. After creating the user account 40, the gateway server 18 returns an information packet to the camera containing new ISP information (if needed), an account ID, and an account password in step 124. The information packet may also contain a default action list specifying what actions should be taken with respect to the images, an advertisement for display on the camera, and the URL of the entity-specific website 22.

It should be noted that if the camera is used in conjunction with an IP direct phone or is provided with a phone number for connected to a dedicated server where the user is not billed separately for the ISP connection, then the steps of providing the camera with default ISP info and returning new ISP info, may be omitted.

The camera receives the information packet in step 126, and writes a configuration file to memory 82a containing the ID, password, and default action list in step 128. The camera then displays the action list on the camera's LCD screen for selection by the user in step 130. The camera may optionally display the user's account information as well.

In an alternative preferred embodiment where security is a concern, the camera 14 first logs off the special ISP and the gateway accounts, and then reconnects using new ISP and gateway accounts in order to retrieve the action lists 48.

FIG. 5 is a block diagram illustrating an example action list 48 displayed on the LCD screen of the camera 14. The action list 48 is shown displaying three major options; printing the uploaded images, saving the uploaded images in

11

the user's shoebox, and sending the images to Mom. Under the printing option, the user may select from various size prints. Rather than nested menu categories as shown under the printing option, the action list 48 may be displayed with each action listed as a separate item (e.g., "Send 4x8 prints to Mom", "Send 5x7 prints to me").

Referring again to FIG. 4B, after the user selects one or more actions from the action list 48, the camera begins to upload the images along with the selected actions in step 132 and displays a progress bar on the screen in step 134. In one preferred embodiment, the camera may also display the advertisement sent in the information packet from the gateway server 18. The advertisement may advertise the controlling entity 12, the entity's photo-sharing site 22, or the photo sharing service 16. After all the images are uploaded and associated with the user's account 40, the camera breaks the connection with the gateway server in step 136. At this point, the camera 14 may also display the URL of the entity-specific website 22 to the user.

The next time the user uploads images, the camera will use any new ISP information received to connect to the Internet and will use the account ID and password written to memory 82a when connecting to the gateway server 18. Thus, by using unique camera information, such as the serial number, to establish a web site account upon first use, the present invention eliminates the need for the user to type in information to establish a web site accounts.

To further explain the present invention from a user interaction point of view, consider the following scenario where a user named Jack has just purchased a digital camera 14 from a store and unpacks the camera from the box. There is a "Quick Start Guide" which guides him in getting started. Jack pops in the batteries, sets the date and time, and takes some pictures of his dog and his new baby. Jack can see the pictures have come out well on the small LCD, and now wants to try sharing them with his parents.

The Quick Start Guide says to select the photos to be sent using the "Select" button, and then press the "Send" button. So, Jack navigates to each of the baby pictures, selects them, and then presses the "Send" button. Instantly, a dialog comes up on the LCD screen: "No Receiving Device Found! Please turn on your phone or other connecting device." Oops! Jack pulls out his cell phone, and turns it on. He presses the "Continue" button. Jack does not see this happen, but the camera now "discovers" the cell phone, and immediately presents another dialog: "4 Images Selected. Press Record to add a Sound Note, or Continue to send". Although Jack finds the proposition of recording sound intriguing, he decides to skip it and presses the "Continue" button. Immediately, the dialog is replaced with a "Connecting . . ." dialog.

Shortly, another dialog appears: "Your camera serial number is 38147. Please write this down. You will need it to access your web photo albums". Jack writes the number down on the spot provided in the Quick Start Guide, and presses OK.

Another "Connecting . . ." dialog appears, and is then quickly replaced by another dialog, which says "A free, temporary account has been set up for you at www.photo-sharing.service.com/new_accounts. You will need your camera serial number to access your photos and complete the setup of your account. Please complete the account setup within 30 days". Jack writes down the URL in the space provided in the Quick Start Guide, and presses the OK button. Jack doesn't know it, but during this dialog, the camera has begun transmitting his images and is already partially complete.

12

A new dialog comes up, with a progress bar. Jack is surprised to see that the transmission is already almost ½ done. Below the progress bar, the dialog says "Press 'Continue' to use camera during photo transmission, or wait for progress bar to complete". Jack is interested in watching how fast his images are transmitted, and decides to watch the progress bar complete. A "Transmission Successful" dialog appears. Jack presses the OK button. The camera returns to the review mode.

Jack is pretty excited—he just sent four baby pictures to the Internet. Jack then decides to see what happened to his images so he turns on his PC. After connecting to the Internet, Jack types in the URL from the Quick Start Guide. A Photo-sharing service web page appears, welcoming Jack to the photo-sharing service. After looking briefly at the welcome page, Jack types in his serial number from the Quick Start Guide, and selects his camera's model number from a pop-up menu. Jack clicks on a "Submit" button on the web page.

Jack now sees a page which shows thumbnails of the baby pictures he just sent, the page is entitled "My Shoebox". The page explains to Jack that he is looking at his on-line digital photo shoebox. Since the server 18 knows this is Jack's first visit, special help messages may appear. Various options are provided via buttons and text links. One that catches Jack's eye is CREATE WEB PHOTO ALBUM. Jack clicks this button, and works his way through the process of setting up an on-line photo album. This includes selecting photos from the shoebox, as well as selecting layout and style. One of the check-box items Jack is offered is "Make this album a camera Action List Item". Jack doesn't know what that is, so he clicks the Action List link, which brings up a brief description "If you check this box, you will be able to send pictures directly from the camera to this photo album!" Jack finds this interesting, so he closes the description window, and checks the box. Jack also enters the email address for his parents and his wife's parents, so they can be notified to come and see his photo album, which he has entitled "Our First Baby".

One of the buttons Jack does not click is the "Complete Account Setup" button. He knows that he has 30 days to do that chore, and figures he will get back to it later.

Jack's wife arrives home from shopping at this point, and Jack wants to show her how the new camera works. He starts by showing her the baby album on the PC, then decides to take some pictures of them holding the baby. Jack then selects the images, and presses the "Send" button again.

Since his cell phone is still on, sitting on the table a few feet away, the connection goes smoothly and quickly. A new dialog pops up, surprising Jack. It says "Select the destination for your pictures" and offers two choices: "My Shoebox" and "Our First Baby". Jack is amazed, he doesn't realize that the server 18 downloaded his action list to his camera during the connection. Jack decides to select the "Our First Baby" web album as the destination for the images, and clicks OK. The pictures are sent as before. After the transmission has completed, Jack goes to the PC to check on the photo album. When Jack refreshes the album page, he now sees the additional pictures he just sent, along with the pictures he sent before.

Jack spots a "Send Prints" link on the web page, and clicks it. He is led through a selection of print types, mailing addresses, and credit card info to make it possible to send prints. He is offered the option of completing the setup of his account. Jack decides to do that now, and proceeds to fill out the requested information, including his credit card number.

13

Once the account setup is complete, Jack continues the print order. One of the radio button items is "Make prints a separate Action List Item" or "Make prints part of your Action List". Jack remembers something about Action Lists from before, but is not sure what this means. The description says "Making a separate action list for prints allows you to decide in the camera to send to the photo album and send prints or to just send to the photo album." Jack thinks this is cool, and clicks the "Make prints a separate Action List" item. The next time Jack sends photos from the camera, his action list will be updated to present three choices: My Shoebox, Our First Baby, and Our First Baby w/Prints.

The underlying technology supporting this scenario are summarized below. Other functions and features are assumed, but not required for this scenario:

1. Two-way connection between camera and portal
2. Camera metadata included in the request
3. If not using an IP direct connection,
 - 3.1. Default ISP connection info built into the camera for the first connection (country specific)
 - 3.2. Downloaded assigned ISP information from the portal to the camera
4. Software capable of recognizing automatically a set of supported phones and adjusting the protocol to match
5. Action Lists maintained on the server that are automatically downloaded to the camera to update the camera selection list each time a connection is made
6. An On-line shoebox
7. Ability to download files to the camera from the server. The files could be text, GIF, animated GIF, JPG, or even a script or applet. This feature enables the ability to display advertisements on the camera, remind the user of remaining time to complete account setup, make special offers, and indicate limits reached.

A method and system for automatically configuring a web-enabled digital camera to access the Internet has been disclosed. Although the present invention has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations to the embodiments and those variations would be within the spirit and scope of the present invention. For example, although the photo-sharing service has been described as including the gateway server and the database, the database may be located elsewhere. Also, the gateway server may be used to control account information, while one or more other servers may be used to provide the web pages of the entity-specific websites. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.

What is claimed is:

1. A method for automatically establishing a user account and for configuring a hand-held image capture device which includes a digital camera for accessing a site on a public network using the user account, the method comprising the steps of:

- (a) establishing a connection to a website server, the website server being responsible for establishing and maintaining website accounts;
- (b) checking for the presence of account information on the image Capture Device and if none are found, determining that this is a first-time connection and automatically sending image capture device information uniquely identifying the electronic device to the website server so that the server can set-up the account information based on the image capture device information;

14

(c) receiving user account information from the server, wherein the user account information includes an account ID and password; and

(d) storing the user account information on image capture device for use a next time the image capture device accesses the website, whereby the user does not have to enter account information in order to establish the connection with the public network or to establish the website account.

2. The method of claim 1 further including the step of providing a digital camera as the hand-held image capture device.

3. The method of claim 2 further including the step of sending at least a partial serial number as the image capture device information uniquely identifying the electronic device.

4. The method of claim 3 further including the step of uploading captured images to the website server.

5. The method of claim 4 wherein step (a) further including the steps of:

- (i) providing the image capture device with default ISP information; and
- (ii) establishing a connection to an ISP using the default ISP information.

6. The method of claim 5 further including the step of terminating the connection with the ISP when the uploading of the images is complete.

7. The method of claim 6 further including the step of providing the digital camera with a default ISP access number, user ID, and password.

8. A system for automatically establishing a user account and for configuring a hand-held image capture device which includes a digital camera image capture device for accessing a site on a public network using the user account, comprising:

means for establishing a connection to a website server, the website server being responsible for establishing and maintaining website accounts;

means for checking for the presence of account information on the image capture device, and if none are found, determining that this is a first-time connection and automatically sending image capture device information uniquely identifying the image capture device to the website server so that the server can set-up the account information based on the image capture device information; and

means for receiving user account information from the server, wherein the user account information includes an account ID; and

means for storing the user account information on image capture device for use a next time the image capture device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account.

9. The system of claim 8 wherein the hand-held image capture device comprises a digital camera.

10. The system of claim 9 wherein the image capture device information uniquely identifying the image capture device includes a serial number of the image capture device.

11. The system of claim 10 wherein the camera uploads captured images to the website server.

12. The system of claim 9 wherein the camera further includes means for providing the image capture device with default ISP information, and means for establishing a connection to an ISP using the default ISP information.

13. The system of claim 12 wherein the digital camera terminates the connection with the ISP upon completion of image uploading.

15

14. The system of claim 13 wherein the camera is provided with a default ISP access number, user ID, and password.

15. A method for automatically establishing a user account and for configuring a digital camera to access a website on a public network using the user account, the digital camera including captured images and communication means for accessing the public network, the method comprising the steps of:

- (e) storing default ISP information on the digital camera, the default ISP information including an access number;
- (f) in response to a user request to upload images to the website, determining whether this is a first-time connection;
- (g) if it is first-time connection, establishing communication with the default ISP by dialing the access number;
- (h) establishing communication with the website and automatically sending a digital camera ID to the website;
- (i) using the digital camera ID to set up a user account on the website;
- (j) sending user account information to the digital camera, the user account including an account ID and an account password;
- (k) storing the user account information on the camera; and
- (l) uploading images to the website and associating the images with the user account, whereby the user does not have to enter information to establish a connection with the public network or to establish the website account.

16. The method of claim 15 further including the step of sending at least a partial serial number as the digital camera ID.

17. The method of claim 16 further including the step of receiving new ISP information from the website.

18. The method of claim 17 further including the step of using the new ISP information a next time the digital camera establishes a connection with the public network.

16

19. A system for automatically establishing a user account and for configuring a digital camera for accessing a website on a public network using the user account, the digital camera including images captured by user and communication means for wirelessly accessing the public network, the system comprising the steps of:

- means for storing default ISP information on the digital camera, the default ISP information including an access number;
- means for determining whether this is a first-time connection in response to a user request to upload images to the website;
- means for establishing communication with the default ISP by dialing the access number;
- means for establishing communication with the website and sending a digital camera ID to the website;
- means for using the digital camera ID to set up a user account on the website;
- means for automatically sending user account information to the digital camera, the user account including an account ID and an account password;
- means for storing the user account information on the camera; and
- means for uploading images to the website and associating the images with the user account, whereby the user does not have to enter information to establish a connection with the public network or to establish the website account.

20. The system of claim 19 wherein the digital camera ID includes a serial number.

21. The system of claim 20 wherein the digital camera receives new ISP information from the website.

22. The system of claim 21 wherein the digital camera uses the new ISP information a next time the digital camera establishes a connection with the public network.

* * * * *

(B) Evidence for Claims on Appeal – Entered by Examiner

The following items listed below are hereby entered as evidence entered by the Examiner. Also listed for each item is where said evidence was entered into the record by the Examiner.

- (1) Copy US Patent 6784924 08-2004 (Ward). This evidence was entered into the record by the Examiner at 23. of the Office Action dated 11/05/2004.
- (2) Copy US Patent 6636259 10-2003 (Anderson). This evidence was entered into the record by the Examiner at 33. of the Office Action dated 11/05/2004.
- (3) Copy US Patent 6353848 03-2002 (Morris). This evidence was entered into the record by the Examiner at 47. of the Office Action dated 11/05/2004.
- (4) Copy US Patent 6662218 12-2003 (Mighdoll). This evidence was entered into the record by the Examiner at 49. of the Office Action dated 11/05/2004.
- (5) Copy US Patent 6636259 02-2004 (Cook). This evidence was entered into the record by the Examiner at 53. of the Office Action dated 11/05/2004.
- (6) Copy US Patent 6442573 08-2002 (Schiller). This evidence was entered into the record by the Examiner at 64. of the Office Action dated 11/05/2004.
- (7) Copy US Patent 6058399 05-2000 (Morag). This evidence was entered into the record by the Examiner at 65. of the Office Action dated 11/05/2004.
- (8) Copy US Patent 5751719 05-1998 (Chen). This evidence was entered into the record by the Examiner at 66. of the Office Action dated 11/05/2004.
- (9) Copy US Patent 6463474 10-2002 (Fuh). This evidence was entered into the record by

the Examiner at 67. of the Office Action dated 11/05/2004.

- (10) Copy US Patent 6714979 03-2004 (Brandt). This evidence was entered into the record by the Examiner at 68. of the Office Action dated 11/05/2004.
- (11) Copy US Patent 6567122 05-2003 (Anderson '122). This evidence was entered into the record by the Examiner at 69. of the Office Action dated 11/05/2004.
- (12) Copy US Patent 6784924 08-2004 (Ward). This evidence was entered into the record by the Examiner at 10. of the Office Action dated 06/30/2005.
- (13) Copy US Patent 6636259 10-2003 (Anderson). This evidence was entered into the record by the Examiner at 20. of the Office Action dated 06/30/2005.
- (14) Copy US Patent 6784924 08-2004 (Ward). This evidence was entered into the record by the Examiner at 34. of the Office Action dated 06/30/2005.
- (15) Copy US Patent 6662218 12-2003 (Mighdoll). This evidence was entered into the record by the Examiner at 36. of the Office Action dated 06/30/2005.
- (16) Copy US Patent 6636259 02-2004 (Cook). This evidence was entered into the record by the Examiner at 40. of the Office Action dated 06/30/2005.
- (17) Copy US Patent 6784924 08-2004 (Ward). This evidence was entered into the record by the Examiner at 3. of the Office Action dated 04/10/2006.
- (18) Copy US Patent 6636259 10-2003 (Anderson). This evidence was entered into the record by the Examiner at 13. of the Office Action dated 04/10/2006.
- (19) Copy US Patent 6353848 03-2002 (Morris). This evidence was entered into the record by the Examiner at 28. of the Office Action dated 04/10/2006.
- (20) Copy US Patent 6662218 12-2003 (Mighdoll). This evidence was entered into the record by the Examiner at 30. of the Office Action dated 04/10/2006.

- (21) Copy US Patent 6636259 02-2004 (Cook). This evidence was entered into the record by the Examiner at 37. of the Office Action dated 04/10/2006.
- (22) Copy US Patent 6784924 08-2004 (Ward). This evidence was entered into the record by the Examiner at 3. of the Office Action dated 02/15/2007.
- (23) Copy US Patent 6636259 10-2003 (Anderson). This evidence was entered into the record by the Examiner at 13. of the Office Action dated 02/15/2007.
- (24) Copy US Patent 6353848 03-2002 (Morris). This evidence was entered into the record by the Examiner at 28. of the Office Action dated 02/15/2007.
- (25) Copy US Patent 6662218 12-2003 (Mighdoll). This evidence was entered into the record by the Examiner at 30. of the Office Action dated 02/15/2007.
- (26) Copy US Patent 6636259 02-2004 (Cook). This evidence was entered into the record by the Examiner at 36. of the Office Action dated 02/15/2007.
- (27) Copy US Patent 6934372 08-2005 (Lynam). This evidence was entered into the record by the Examiner at page 14 at A of the Office Action dated 02/15/2007.
- (28) Copy US Patent 6636259 10-2003 (Anderson). This evidence was entered into the record by the Examiner at 7. of the Office Action dated 05/30/2008.
- (29) Copy US Patent 6353848 03-2002 (Morris). This evidence was entered into the record by the Examiner at 31. of the Office Action dated 05/30/2008.
- (30) Copy US Patent 6662218 12-2003 (Mighdoll). This evidence was entered into the record by the Examiner at 33. of the Office Action dated 05/30/2008.
- (31) Copy US Patent 6636259 02-2004 (Cook). This evidence was entered into the record by the Examiner at 35. of the Office Action dated 05/30/2008.
- (32) Copy US Patent 6636259 10-2003 (Anderson). This evidence was entered into the

record by the Examiner at 3. of the Office Action dated 06/30/2009.

(33) Copy US Patent 6353848 03-2002 (Morris). This evidence was entered into the record by the Examiner at 29. of the Office Action dated 06/30/2009.

(34) Copy US Patent 6662218 12-2003 (Mighdoll). This evidence was entered into the record by the Examiner at 31. of the Office Action dated 06/30/2009.

(35) Copy US Patent 6636259 02-2004 (Cook). This evidence was entered into the record by the Examiner at 33. of the Office Action dated 06/30/2009.

(36) Copy US Patent 6934372 08-2005 (Lynam). This evidence was entered into the record by the Examiner at 46. of the Office Action dated 06/30/2009.

Copies of all References follows.

//



US006784924B2

(12) United States Patent
Ward et al.**(10) Patent No.: US 6,784,924 B2****(45) Date of Patent: Aug. 31, 2004****(54) NETWORK CONFIGURATION FILE FOR
AUTOMATICALLY TRANSMITTING
IMAGES FROM AN ELECTRONIC STILL
CAMERA****(75) Inventors:** Joseph Ward, Hilton, NY (US);
Kenneth A. Parulski, Rochester, NY
(US); James D. Allen, Rochester, NY
(US)**(73) Assignee:** Eastman Kodak Company, Rochester,
NY (US)**(*) Notice:** Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 0 days.**(21) Appl. No.: 09/004,046****(22) Filed: Jan. 7, 1998****(65) Prior Publication Data**

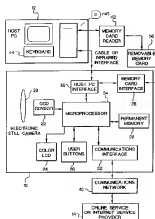
US 2003/0142215 A1 Jul. 31, 2003

Related U.S. Application Data**(60)** Provisional application No. 60/037,963, filed on Feb. 20,
1997, and provisional application No. 60/037,962, filed on
Feb. 20, 1997.**(51) Int. Cl.** **H04N 5/225****(52) U.S. CL.** **348/207.1; 348/211.3****(58) Field of Search** **348/207, 211,**
348/212, 213, 231, 232, 233, 552, 239,
211/99-211.13, 211.3, 17; 710/62, 3, 8;
345/327; 386/48, 117; 725/131-133, 139-141,
151-153, 709/217-219, 220, 705/26, 27,
455/566**(56) References Cited****U.S. PATENT DOCUMENTS**4,524,381 A 6/1985 Konishi
4,574,319 A 3/1986 Konishi
4,825,457 A 4/1989 Lebowitz
5,241,659 A 8/1993 Parulski et al.
5,477,264 A 12/1995 Sarbadhikan et al.5,606,365 A 2/1997 Maurinus et al.
5,663,678 A 9/1997 Chang
5,666,159 A 9/1997 Parulski et al.
5,737,491 A * 4/1998 Allen et al. 704/270
5,800,005 A * 9/1998 Hull et al. 455/566
5,825,432 A * 10/1998 Yonezawa 348/563
6,111,604 A * 8/2000 Hashimoto et al. 348/220
6,122,005 A * 9/2000 Sasaki et al. 348/211.3
6,166,729 A * 12/2000 Acosta et al. 345/719
6,449,001 B1 * 9/2002 Levy et al. 348/1404**OTHER PUBLICATIONS**User's Manual—Axis 2420 Network Camera, 1996 (Axis
Communications).*U.S. patent application Ser. No. 60/037,963, Parulski et al.
“The Universe of Smart Cards: EasySIM software”,
Schlumberger Limited.

“Claris Emailer” from Claris’ home page on internet.

“Kodak Professional Digital Camera System—User’s
Manual,” Eastman Kodak Company, 1991–1992, pp. x to xv,
2–1 to 2–3, 3–29 to 3–30, and 4–1 to 4–49.

* cited by examiner

Primary Examiner—Tuan Ho**(74) Attorney, Agent, or Firm—Pamela R. Crocker****(57) ABSTRACT**A network configuration file is generated at a host computer
and downloaded to a digital camera. This file contains
instruction information for communicating with a selected
destination via a communications interface. The digital
camera includes a “send” button or LCD icon which allows
the user to easily transmit one or more images via a wired
or wireless communications interface to a desired
destination, which among other possibilities may be an
Internet Service Provider or a digital photofinishing center.
When the user selects this option, the communications port
settings, user account specifics, and destination connection
commands are read from the network configuration file on
the removable memory card. Examples of these settings
include serial port baud rate, parity, and stop bits, as well as
account name and password.**9 Claims, 4 Drawing Sheets**

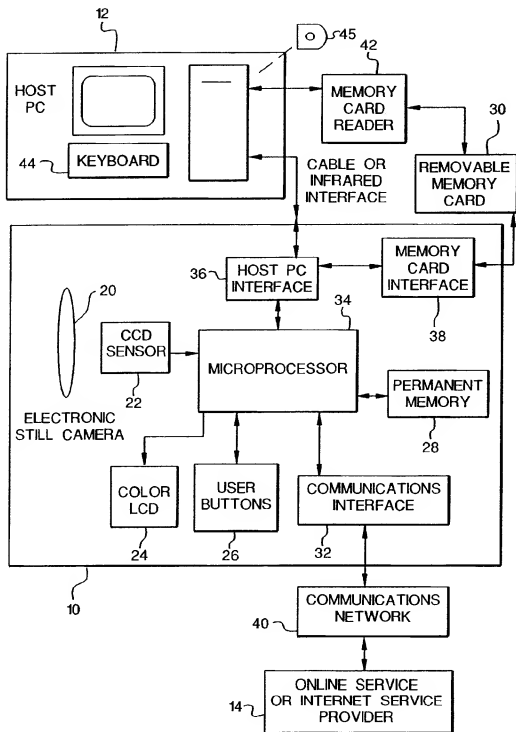
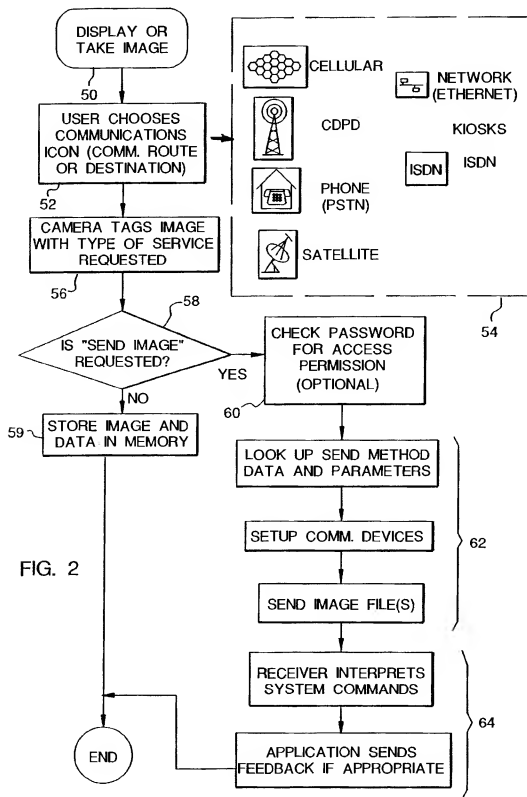


FIG. 1



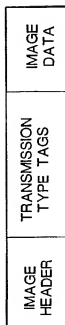


FIG. 3



FIG. 4

FIG. 4A

PHONE (PUBLIC SWITCHED TELEPHONE NETWORK)



PROTOCOL TYPE	PHONE NUMBER	DEFAULT SETTINGS	MODERN CONTROL STRING	ACCOUNT DATA	PASSWORD	SYSTEM COMMANDS
------------------	-----------------	---------------------	-----------------------------	-----------------	----------	--------------------

CELLULAR OR PCS

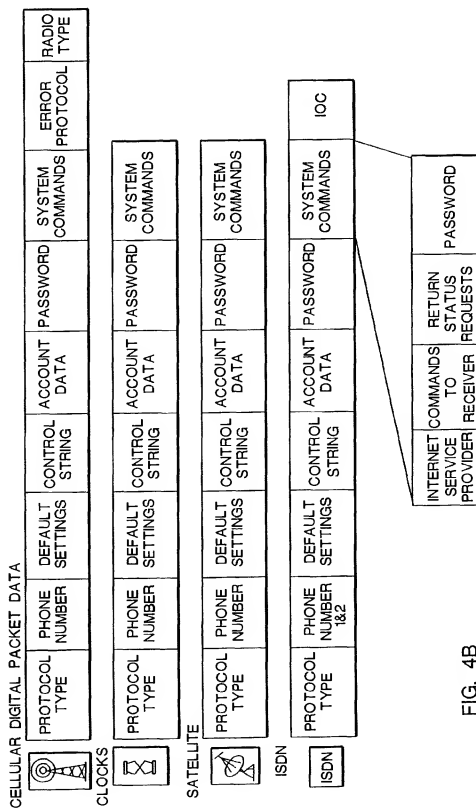


PROTOCOL TYPE	PHONE NUMBER	DEFAULT SETTINGS	MODERN CONTROL STRING	ACCOUNT DATA	PASSWORD	SYSTEM COMMANDS	ERROR PROTOCOL	RADIO TYPE
------------------	-----------------	---------------------	-----------------------------	-----------------	----------	--------------------	-------------------	---------------

WIRELESS LAN



PROTOCOL TYPE	PHONE NUMBER	DEFAULT SETTINGS	PARA- METER FILE	ACCOUNT DATA	PASSWORD	SYSTEM COMMANDS
------------------	-----------------	---------------------	------------------------	-----------------	----------	--------------------



1

NETWORK CONFIGURATION FILE FOR AUTOMATICALLY TRANSMITTING IMAGES FROM AN ELECTRONIC STILL CAMERA

CROSS-REFERENCE TO RELATED APPLICATION(S)

THIS APPLICATION CLAIMS BENEFIT OF PROVISIONAL APPLICATION SERIAL NO. 60/037,962 FILED Feb. 20, 1997.

Reference is made to commonly assigned copending applications Serial No. 60/037,963, filed Feb. 20, 1997 entitled "Electronic Camera with 'Utilization' Selection Capability" and filed in the names of Kenneth A. Parulski, Joseph Ward, and Michael C. Hopwood, which is assigned to the assignee of this application.

FIELD OF THE INVENTION

The invention relates generally to the field of photography, and in particular to electronic photography. More specifically, the invention relates to a digital camera that interfaces with a host computer.

BACKGROUND OF THE INVENTION

Digital cameras, such as the Kodak Digital Science DC25™ camera, allow images to be utilized on a home computer (PC) and to be incorporated into e-mail documents and personal home pages on the World Wide Web. Presently, images must be copied to the PC and transmitted as e-mail, for example using an online service or an Internet Service Provider (ISP), via a modem from the user's PC. It would be desirable to be able to transmit pictures directly from the digital camera instead of first transferring the pictures to a PC. For instance, on a vacation trip, it is desirable to immediately share pictures with friends or relatives via e-mail or Internet access. It is also desirable to transmit pictures from a location without PC access in order to free up camera storage to take additional pictures. There are a wide variety of connection means to online services such as America On Line, ISPs, and bulletin board services. Each of these services typically requires an account name and password, as well as local telephone access numbers, and specific communications settings. It would be difficult to provide an easy-to-use means with buttons or menus on a small digital camera to input and/or modify all of these required settings.

SUMMARY OF THE INVENTION

The present invention is directed to overcoming one or more of the problems set forth above. Briefly summarized, according to one aspect of the present invention, a network configuration file is generated at a host computer and downloaded to a digital camera. This file contains instruction information for communicating with a selected destination via a communications interface. The digital camera includes a "send" button or LCD icon which allows the user to easily transmit one or more images via a wired or wireless communications interface to a desired destination, which among other possibilities may be an Internet Service Provider or a digital photofinishing center. When the user selects this option, the communications port settings, user account specifics, and destination connection commands are read from the network configuration file. Examples of these settings include serial port baud rate, parity, and stop bits, as well as account name and password.

2

In addition, information about which image or images to transmit is entered using the user buttons on the digital camera. This information is used to automatically establish a connection, log-in to the desired destination, and to transmit the image. The transmission may occur immediately after the pictures are taken, for example if the camera has a built-in cellular phone modem, or at a later time, when the camera is connected to a separate unit (such as a dock, kiosk, PC, etc.) equipped with a modem. In the latter case, a "utilization file" is created to provide information on which images should be transmitted to which account.

These and other aspects, objects, features and advantages of the present invention will be more clearly understood and appreciated from a review of the following detailed description of the preferred embodiments and appended claims, and by reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a system block diagram of the invention.
FIG. 2 is a diagram showing the steps used to automatically transmit images using the network configuration file.
FIG. 3 is a diagram of an image file.
FIG. 4 is a diagram showing several versions of the network configuration file.

DETAILED DESCRIPTION OF THE INVENTION

Because imaging systems and devices are well known, the present description will be directed in particular to elements forming part of, or cooperating more directly with, apparatus in accordance with the present invention. Elements not specifically shown or described herein may be selected from those known in the art. Some aspects of the present description may be implemented in software. Unless otherwise specified, all software implementation is conventional and within the ordinary skill in the programming arts.

A system block diagram of the invention is shown in FIG. 1 including an electronic still camera 10, a host computer (PC) 12 and a service provider 14. The camera includes an optical section 20 for imaging a scene upon a CCD sensor 22 and generating an image signal, a liquid crystal display (LCD) 24 for displaying images and other information, a number of user input buttons 26, both permanent memory 28 and removable memory 30, and an internal communications interface 32 (e.g., modem). This interface may connect to a variety of known networks, such as a public switched telephone network (PSTN), ISDN, an RF cellular phone network, or Ethernet. The camera 10 also includes a microprocessor 34 for generally controlling the camera functions, as well as the interchange of data with the host PC 12 and the memory card 30 through a host PC interface 36 and a memory card interface 38, respectively. Besides the host PC 12, the system includes a network connection 40 to the online service or ISP (Internet Service Provider) 14. Alternately, the network 40 can connect to the user's home PC 12.

When the camera 10 is first purchased (or at any time thereafter), it is connected to the PC 12 via the host PC 36 interface and a software application (stored on a disc 45) running on the host PC 12 will enable the user to specify the name of a destination ISP or online service and to input from the host PC keyboard 44 the appropriate communication settings and account information. This information generates a network configuration file, which then can be downloaded to the camera 10 through the host PC interface

3

4

36, which may be a wired or infrared (e.g., IrDA) interface, and written to the camera's internal memory 28 and/or the removable memory card 30. Alternatively, a host PC equipped with a memory card reader/writer 42 can write the information directly to the card 30 without connecting the camera through its host PC interface 36. Also, this information could be predetermined by the user and stored in a "preferences" file on the host PC 12 and then transferred to the camera 10 from this file without further intervention by the user. Multiple sets of destination services can be stored on the memory card 30. Typically, keyword or graphic descriptors (e.g., icons) accompany the information in the network configuration file about destination services to enable easy access by the camera user.

The steps used to automatically transmit images using the network configuration file are shown in FIG. 2. After disconnecting the camera from the host PC, the user operates the camera to take pictures (step 50). This is typically done at a remote location, for example while traveling to another city. As the user takes or reviews images on the image LCD display, the decision can be made to transmit one or more images (step 52). This is done by choosing one of the keywords or icons in a menu 54 shown in FIG. 2, which are displayed on the LCD 24 and selected, e.g., through the user buttons 26. (Note that a camera will typically only include a subset (only those desired by the user) of all the different services shown.) The selected image files may be tagged with a code (step 56) indicating which service is requested, as shown in FIG. 3. (Alternately, an "image utilization" file can be created in the camera storing a list of images to be transmitted by a particular method, as described in the cross-referenced depending patent application (U.S. Serial No. 60/037,963). As described in that patent application, the details of an order, e.g., number of print copies to be made from an image and the size of the prints and/or a list of images to be e-mailed to various recipients, is written into the "utilization" file, which identifies the order and includes pointers to the image files that store the images required to "fulfill" the order. The "utilization" file is stored in the internal memory 28 or the memory card 30.)

Next, the system determines whether a request exists to send an image (step 58). If no request is present, the image and associated data is stored in either permanent memory 28 or the memory card 30 (step 59). (Typically, all images are initially saved in memory whether eventually sent or not.) Otherwise, if there is a request to send an image, the user ensures that the camera is connected to the appropriate service (wired telephone line, cellular phone, kiosk, etc.) and pushes a "send" button in the user button section 26, or selects a "send" menu option on the LCD 24. The camera then utilizes the appropriate network configuration file, shown in FIG. 4. Each network configuration file contains items such as the protocol type, phone number, etc., as described in Appendix I. The user password may be checked against the password in the network configuration file to ensure that the user is authorized to connect the camera to the desired service (step 60). Alternately, the stored password in the appropriate configuration file can be used. Next, the camera uses the parameters in the configuration file to establish communications with the service and send one or more image files as selected by the user (steps 62). The service receiver interprets the system commands issued by the camera from the network configuration file list and sends appropriate feedback (such as "transfer in progress" and "transfer complete") which are interpreted by the camera and displayed on the LCD 24 (steps 64).

For example, when the camera uses a normal wired telephone (Public Switched Telephone Network) connection

(i.e., network 40) to the camera's internal modem 32, after the user selects the images to be sent and presses the "send" button, the camera performs the following steps without user intervention:

- 1) Read the appropriate connection parameters from the network configuration file (on the memory card 30 or internal camera memory 28), dial the phone and establish the connection to the destination service 14.
- 2) Read the user's account name and password and transmit these to "log-on" to the service 14.
- 3) Using the appropriate communications protocol (FTP, mailto, etc.), transmit the selected image or images to the destination service 14.

The invention has been described with reference to a preferred embodiment. However, it will be appreciated that variations and modifications can be effected by a person of ordinary skill in the art without departing from the scope of the invention.

Appendix I

These are descriptions of the tags listed in the previous drawing:

Protocol Type

Each communication method has its own protocol, or rules to communicate. This tag identifies that protocol and where to find it. For example, the Network may use TCP/IP and a modem may use XModem.

Phone Number

This is the number of the receiving service. If internet access is requested, this could be the number of the Internet Service Provider. For ISDN, some systems require two phone numbers, dialed and connected to in sequence.

Default Settings

Standard settings that make the communications device compatible with the imaging device.

Modem Control String

Modem and communications devices have a command language that can set them up before they are used. For example, modems have many options controlled by command strings including volume level, the amount of time the carrier is allowed to fail before the system hangs up, and so on.

Account Data

This can be internet account data, charge number data, phone card data, billing address, and data related to the commerce part of the transmission.

Password

Any password needed to get into the communications system. Other passwords to get into the remote application or destination are located in the System Commands section.

System Commands

These are commands that control the end destination.

Error Protocol

In cellular and some other wireless communications, error protocols are used to increase the robustness of the link. For example, MNP10 or ETC may be used for cellular links.

Radio Type

The type of radio used for this communications feature may be identified here. Some cell phones have modems built in, others will have protocols for many communications functions built in. The radio type will make the imaging device adapt to the correct interface.

IOC

ISDN Ordering Code identifies what features are available on the ISDN line provided by the telco. It is used to establish the feature set for that communications link.

5

Internet Service Provider

This identifies the actual service provider and any specific information or sequence of information that the service wants to see during connection and logoff. It also tells the device how to handle the return messages, like "time used" that are returned by the server.

Commands to Receiver

This may be a list of commands to control the receiving application. For example, a command to print one of the images and save the data to a particular file on a PC may be embedded here.

Return Status Requests

This tag can set up the ability of the application to tell if an error has occurred, or what the status of the application might be. The data here will help the device decide if it should continue communicating and a set user interface response can be developed around this feedback.

What is claimed is:

1. A method for directing the automatic transmission of images from an electronic camera, said method including the steps of:

storing network configuration information in a memory of the electronic camera;

using the network configuration information to connect the camera to an online service or internet service provider (ISP);

transferring pictures and data from the camera to the online service or ISP;

receiving information from the online service or ISP indicating that the transfer of the pictures and data is complete; and

displaying information on a display of the electronic camera indicating that the transfer of the pictures and data is complete;

wherein the network configuration information is generated at least in part in a host device adaptable for communication with the electronic camera, based on information previously stored in the host device and utilizable to connect the host device to the online service or ISP.

2. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes a PSTN (Public Switched Telephone Network).

3. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes an ISDN network.

4. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes an RF cellular phone network.

6

5. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes an Ethernet connection.

6. The method of claim 1 wherein the information previously stored in the host device comprises information stored in a network configuration preference file of the host device.

7. A method for directing the automatic transmission of images from an electronic camera, said method including the steps of:

storing network configuration information in a memory of the electronic camera, the network configuration information comprising information for using a communications network to establish a connection between the electronic camera and an online service or internet service provider (ISP); and

in response to user activation in the electronic camera of a transmission command specifying one or more pictures for transmission, automatically performing without further user intervention the steps of:

retrieving the stored network configuration information;

utilizing the retrieved information to establish a connection between the electronic camera and the online service or ISP;

transferring the one or more pictures from the electronic camera to the online service or ISP;

receiving information from the online service or ISP indicating that the transfer of the one or more pictures is complete; and

displaying the information on a display of the electronic camera indicating that the transfer of the one or more pictures is complete;

wherein the network configuration information is generated at least in part in a host device based on information previously stored in the host device, and the generated network configuration information is subsequently downloaded from the host device into the electronic camera in conjunction with the storing step.

8. The method of claim 7 wherein the host device comprises a personal computer.

9. The method of claim 7 wherein the information previously stored in the host device comprises information stored in a network configuration preference file of the host device.

* * * * *



US006636259B1

(12) **United States Patent**
Anderson et al.

(10) **Patent No.:** **US 6,636,259 B1**
(45) Date of Patent: **Oct. 21, 2003**

- (54) **AUTOMATICALLY CONFIGURING A WEB-ENABLED DIGITAL CAMERA TO ACCESS THE INTERNET**
- (75) Inventors: **Eric C. Anderson**, San Jose, CA (US);
Robert Paul Morris, Raleigh, NC (US)
- (73) Assignee: **IPAC Acquisition Subsidiary I, LLC**,
Peterborough, NH (US)
- (*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 32 days.
- (21) Appl. No.: **09/625,824**
- (22) Filed: **Jul. 26, 2000**
- (51) Int. Cl.⁷ **H04N 5/232**
- (52) U.S. Cl. **348/211.3; 348/211.12; 348/14.04**
- (58) **Field of Search** **348/14.01, 14.02, 348/14.04, 14.08, 14.09, 552, 143, 156, 157, 211.3, 211.12; 709/322**
- (56) **References Cited**

U.S. PATENT DOCUMENTS

- 5,430,827 A * 7/1995 Rissanen 704/272
5,737,491 A * 4/1998 Allen et al. 704/270
5,905,736 A * 5/1999 Ronen et al. 370/546
6,064,671 A * 5/2000 Killian 370/389
6,067,571 A * 5/2000 Igarashi et al. 709/232

- 6,167,469 A * 12/2000 Safai et al. 710/62
6,226,752 B1 * 5/2001 Gupta et al. 713/201
6,269,481 B1 * 7/2001 Perlman et al. 717/11
6,502,195 B1 * 12/2002 Colvin 713/202
- * cited by examiner

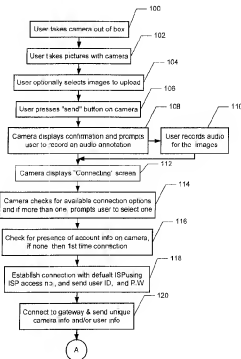
Primary Examiner—Wendy R. Garber
Assistant Examiner—Jacqueline Wilson

(74) *Attorney, Agent, or Firm*—Sawyer Law Group LLP

(57) **ABSTRACT**

The present invention provides a method and system for automatically establishing a user account and for configuring a hand-held electronic device for accessing a site on a public network using the user account. The method and system include establishing a connection to a website server, which is responsible for establishing and maintaining website accounts. The presence of account information on the electronic device camera is then checked, and if none are found, it is determined that this is a first-time connection and information uniquely identifying the electronic device is sent to the website server so that the server can set-up the account information based on the electronic device information. The server then sends user account information to the device, including an account ID and password. The user account information is then stored on electronic device for use the next time the electronic device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network.

22 Claims, 6 Drawing Sheets



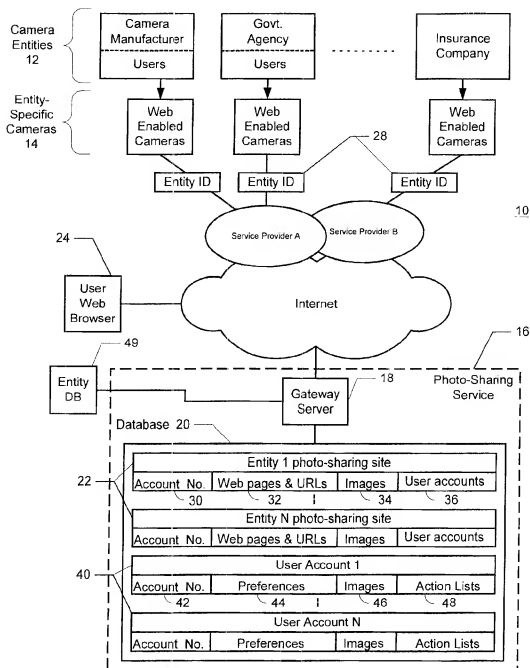


FIG. 1

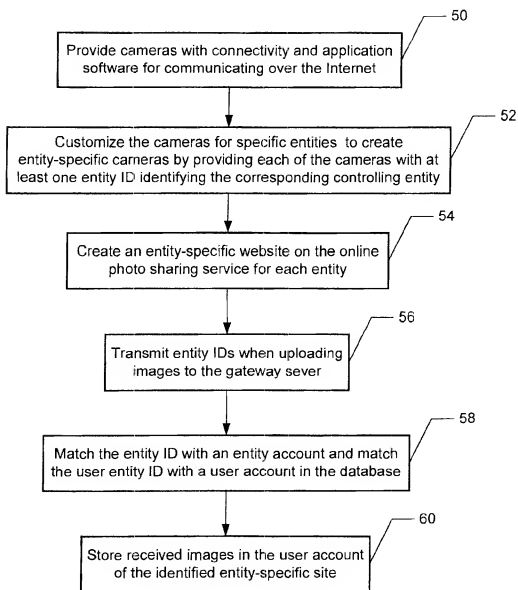


FIG. 2

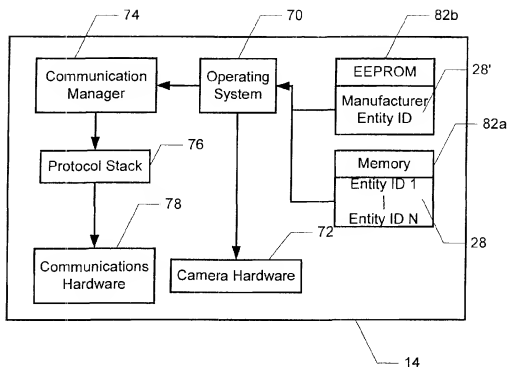


FIG. 3

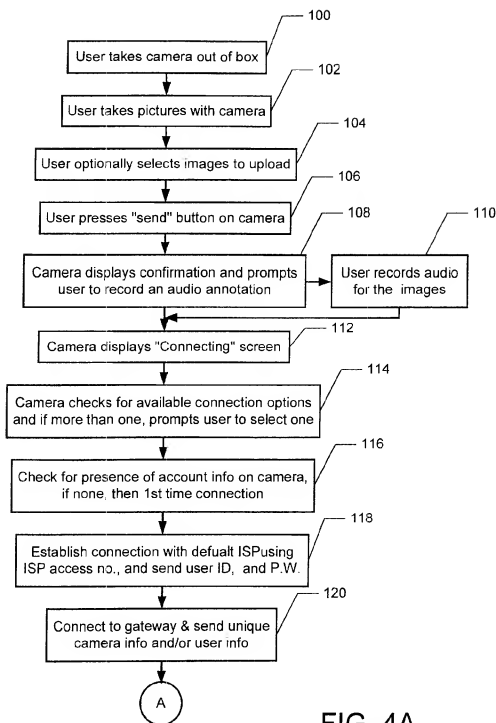


FIG. 4A

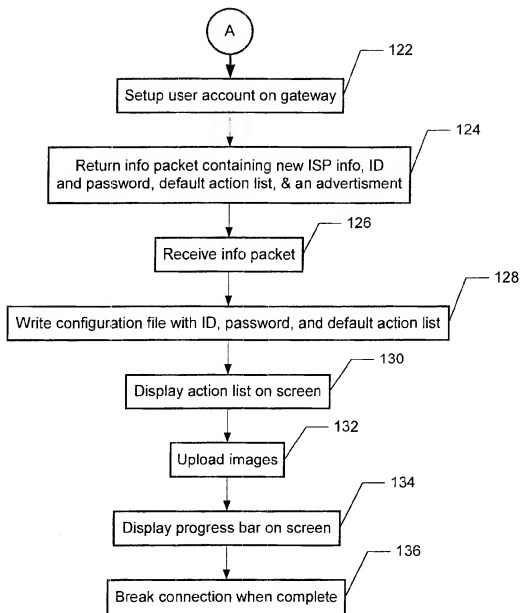


FIG. 4B

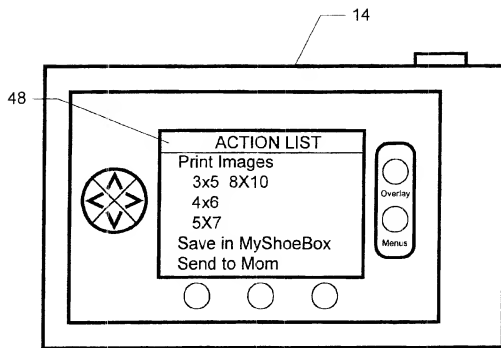


FIG. 5

1

AUTOMATICALLY CONFIGURING A WEB-ENABLED DIGITAL CAMERA TO ACCESS THE INTERNET

CROSS-REFERENCE TO RELATED APPLICATIONS

The present invention is related to co-pending U.S. patent application Ser. No. 09/625,398 entitled "Method and System For Hosting Entity-Specific Photo-Sharing Websites For Entity-Specific Digital Cameras"; and to co-pending U.S. patent application Ser. No. 09/626,418, entitled "Method And System For Selecting Actions To Be Taken By A Server When Uploading Images," which are assigned to the assignee of the present application and filed on the same date as the present application.

FIELD OF THE INVENTION

The present invention relates to a method and system for customizing digital cameras to upload images to an entity-specific photo-sharing websites, and more particularly to automatically configuring a web-enabled digital camera to access the Internet.

BACKGROUND

As the popularity of digital cameras grows, the desire of digital camera users to share their images with others will also continue to grow. New digital camera owners typically try to share their images based on the paradigm of film cameras, in which images are printed on paper and then placed into a photo album. The most straightforward approach to do this with a digital camera is to connect the digital camera directly to a printer to create the prints, and then manually insert the images into a photo album. Users often find this process somewhat complicated and restrictive because standard printers can only print images in limited sizes and requires particular types of paper. And even after the photo album has been assembled, the printed images are not easily shared with many people.

The best approaches to photo-sharing take advantage of the Internet. One such approach is for users to store the digital images on a PC and then send the images to others using email. Several Internet companies now offer an even more convenient approach by providing photo-sharing websites that allow users to store their images for free and to arrange the images into web-based photo albums. Once posted on a photo-sharing website, others may view the images over the Internet.

While convenient for storing digital images, getting the images to the photo-sharing websites can be challenging for users. Most commonly, users must upload their images from the digital camera to a PC using a cable or IrDA, or by inserting the camera's flash card into the PC. From the PC, the user logs onto the Internet and uploads the images to a photo-sharing website. After uploading the images, the user works on the website to arrange the images into web albums and to add any textual information.

Although today's approach for storing images from a digital camera onto a web photo-sharing website and for creating web photo albums works reasonably well, two problems remain that hinder the mainstream adoption of web-based photo-sharing. One problem is that this approach requires the use of a PC, notebook computer, or PDA. While many digital camera users today have PC's, most of those users are early adopters of technology. There are many other consumers who would purchase a digital camera, but are

2

reluctant to do so because they do not yet own a PC or are intimidated by them.

In an effort to address this problem, the assignee of the present application developed an approach to uploading images to the web that doesn't require the use of a PC. In this approach, an email software application is loaded into a digital camera capable of running software that allows the user to e-mail the images directly from the camera. The user simply connects his or her digital camera to a cellphone or modem, runs the e-mail application, and selects the desired images and the email recipients. The selected images are then sent to the recipients as e-mail attachments.

Although emailing photos directly from the camera allows users who do not own a PC to share images over the Internet, these users must still establish accounts with both an Internet service provider (ISP) and the photo-sharing website before being able to post their images. These accounts must also be set-up by PC users as well. For techno savvy users who use a PC to upload the images to the photo-sharing website, establishing the accounts may not be a bother, but even these users may not always have their PC's handy, such as when on vacation, for instance. And for non-PC users, establishing the accounts by entering account information on the digital camera itself may prove to be a cumbersome, if not a daunting, task.

Accordingly, what is needed is an improved method for uploading images from a digital camera to a photo-sharing website on the Internet. In order for online photo-sharing to become more mainstream, an approach that doesn't require a PC or PC expertise and that reduces complexity for the user is required. The present invention addresses such a need.

SUMMARY OF THE INVENTION

The present invention provides a method and system for automatically establishing a user account and for configuring a hand-held electronic device for accessing a site on a public network using the user account. The method and system include establishing a connection to a website server, which is responsible for establishing and maintaining website accounts. The presence of account information on the electronic device camera is then checked, and if none are found, it is determined that this is a first-time connection and information uniquely identifying the electronic device is sent to the website server so that the server can set-up the account information based on the electronic device information. The server then sends user account information to the device, including an account ID and password. The user account information is then stored on electronic device for use the next time the electronic device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network.

According to the method and system disclosed herein, a user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network with the electronic device.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG 1 is a block diagram illustrating an online photo-sharing system in accordance with a preferred embodiment of the present invention.

FIG 2 is a flow chart illustrating a business method for hosting entity-specific photo-sharing websites for entity-specific image capture devices in accordance with a preferred embodiment of the present invention.

3

FIG. 3 is a diagram showing a preferred embodiment of the connectivity and application software of the camera.

FIGS. 4A and 4B illustrate a flow chart of a process for automatically configuring a web-enabled digital camera to access the Internet in a preferred embodiment of the present invention.

FIG. 5 is a block diagram illustrating an example action list 48 displayed on the LCD screen of the camera.

DETAILED DESCRIPTION

The present invention relates to an automatic system for uploading images from a digital camera to entity-specific photo-sharing websites and for automatically establishing accounts. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

FIG. 1 is a block diagram illustrating an online photo-sharing system 10 in accordance with a preferred embodiment of the present invention. The system includes a plurality of camera controlling entities 12 that produce, own, or otherwise control a set of digital cameras 14, and an online photo-sharing service 16. The online photo-sharing service 16 includes a gateway server 18 and an entity/account database 20. The various camera controlling entities 12 contract with the photo-sharing service 16 to transparently host customized photo-sharing websites 22 for each entity, which are referred to herein as entity-specific photo-sharing websites 22. The entity-specific photo-sharing websites 22 are each accessible to the user through the entity's existing Internet site (not shown), and thus appear to users as though the entity-specific photo-sharing websites 22 are hosted by the corresponding entity. According to a preferred embodiment of the present invention, the cameras 14 for a particular entity are customized for that entity to create entity-specific cameras 14, such that when the cameras 14 connect to the Internet, the cameras 14 automatically upload their images to the photo-sharing website of the corresponding entity. In a further aspect of the present invention, the photo-sharing service 16 automatically stores the images in a web album, which is viewable over the Internet by a user's web browser 24.

As used herein, a camera controlling entity 12 is any entity that makes, owns, sells, or controls digital cameras 14, and therefore includes, camera manufacturers, companies, retailers, and end-users. One or more combination of these entities 12 may either contract with the photo-sharing service 16 to provide entity-specific websites 22 for their cameras 14, or have entity information transmitted to the photo-sharing service 16 from the cameras 14. Therefore, a camera controlling entity 12 may include a single entity 12 or a hierarchical relationship of entities 12.

An example of a single entity 12 includes an insurance company that contracts with the photo-sharing service 16 to have all digital cameras 14 used by their agents to transmit their images to a customized insurance photo-sharing website. Examples of a hierarchical relationships of entities 12 includes a camera manufacturer, such as Nikon, that contracts with the photo-sharing service 16 to have all Nikon digital cameras 14 transmit their images to the customized

4

Nikon photo-sharing website. Since the images of different users must be distinguished, each user of a Nikon camera 14 would also constitute an entity within the Nikon website so that the images from different users can be distinguished. Other examples of hierarchical entity relationships include a retailer and its consumers, a real estate agency and its agents, community groups and its members, and government agencies and its employees, for instance.

In a preferred embodiment, the cameras 14 are customized for their respective entities 12 by providing the cameras 14 with software for transmitting entity ID information 28 identifying its controlling entity 12 to the photo-sharing service 16. The photo-sharing service 16 in conjunction with the gateway server 18 and the entity/account database 20 hosts the entity-specific photo-sharing websites 22. Each entity-specific website 22 is identified in the database 20 by an entity account number 30 and includes the web pages and URIs 32 comprising the website, the images and web albums 34 stored on the website, and the user account numbers 36 of authorized users. The database 20 also includes user accounts 40, each of which comprises a user account number 42, user preferences 44, the user's images 46, and action lists 48, explained further below.

The gateway server 18, which communicates with the cameras 14 during image uploading, receives one or more entity IDs 28 from each camera 14 and matches the entity ID 28 with an entity account 30 in the database 20. The images are then automatically associated with the photo-sharing website 22 of the identified entity 12 and/or the identified user.

After the images are uploaded, a user of the camera 14 may visit the online photo-sharing website 22 over the Internet to view the images via a web browser 24. Since the photo-sharing websites 22 are transparently hosted by the photo-sharing service 16, each photo-sharing website 22 appears as though it is hosted by the entity itself, rather than the third party service.

In one embodiment, the cameras 14 may connect to the Internet via a service provider 26, which may include a wireless carrier and/or an Internet service provider (ISP) that is capable of servicing many devices simultaneously. In a preferred embodiment, each of the cameras 14 is provided with wireless connectivity for connecting to the Internet, and are therefore so called "web-enabled" devices, although a wired connection method may also be used.

The cameras 14 may be provided with wireless connectivity using anyone of a variety of methods. For example, a cellphone may be used to provide the digital camera 14 with wireless capability, where the camera 14 is connected to the cellphone via a cable or some short-range wireless communication, such as Bluetooth. Alternatively, the camera 14 could be provided with built-in cellphone-like wireless communication. In an alternative embodiment, the digital camera 14 is not wireless, but instead uses a modem for Internet connectivity. The modem could be external or internal. If external, the camera 14 could be coupled to modem via any of several communications means (e.g., USB, IEEE1394, infrared link, etc.). An internal modem could be implemented directly within the electronics of camera 14 (e.g., via a modem ASIC), or alternatively, as a software-only modem executing on a processor within camera. As such, it should be appreciated that, at the hardware connectivity level, the Internet connection can take several forms. Hence, it should be appreciated that the present invention is not limited to any particular method of accessing the Internet.

5

In a preferred embodiment, the entity-specific websites 22 are customized to seamlessly integrate into the entity's existing website by following the look and feel of the entity's existing website. The entity-specific websites 22 are hosted on the photo-sharing service 16, but a link to the entity-specific websites 22 may be provided on the homepage of the corresponding entity's existing website. Thus, in order to view a web album on an entity-specific website 22, the user must visit the entity's existing website and click the link to the entity-specific website 22, where the user's browser 24 will be transparently directed to the photo-sharing service 16 and be provided with the web pages 32 of the entity-specific website 22.

As an example of the operation of the photo sharing system 10, consider the following scenario. Assume that Minolta and Nikon are entities 12 that have contracted with the photo-sharing service 16, and that the photo-sharing service 16 hosts a photo-sharing website 22 for Minolta and a photo-sharing website 22 Nikon. The Minolta cameras 14 would be provided the entity ID 28 for Minolta and the Nikon cameras 14 would be provided the entity ID 28 for Nikon. When the Minolta and the Nikon cameras 14 send sets of images to the photo-sharing service 16, the gateway server 18 would distinguish the cameras 14 by the entity IDs 28 and would direct the set of images received from Minolta cameras 14 to Minolta's photo-sharing website, and would direct the images from Nikon cameras 14 to Nikon's photo-sharing website. To view the images, the owners of the cameras 14 would use a browser 24 on their PC or PDA to visit the URL of the Minolta or Nikon photo-sharing websites 22. In one preferred embodiment, the photo-sharing service 16 sends the URL of the entity-specific website 22 directly to the camera 14 for display to inform the user of the address.

According to the present invention, the photo-sharing service 16 provides business-to-business and business-to-consumer business models. The service is business-to-business because the service provides companies, such as camera manufacturers, with a complete end-to-end solution for their cameras 14. The solution includes customized software for their cameras 14 for sending images over the internet, and an internet website for storing images from those cameras 14 on a branded website that appears to be hosted by the company. The service is business-to-consumer because it allows users of digital cameras 14 with an automatic solution for uploading captured images from a digital camera 14 to an online photo-sharing website, without a PC.

According to one preferred embodiment of the present invention, the photo-sharing service 16 provides a method of doing business whereby the photo-sharing service 16 shares revenue based on the hierarchical relationship of the entities 12. For example, if the photo sharing service 16 charges a fee for receiving and/or storing the images received from the entity-specific cameras 14, then the photo sharing service 16 may share a portion of the fee with the manufacturer and/or third party supplier of the camera 14 that uploaded the images, for instance. Revenue may also be shared with the wireless service provider providing the connection with the photo sharing service 16.

FIG. 2 is a flow chart illustrating a business method for hosting entity-specific photo-sharing websites for entity-specific image capture devices, such as digital cameras, in accordance with a preferred embodiment of the present invention. First, the cameras 14 are provided with connectivity and application software for communicating over the Internet in step 50. In a preferred embodiment, this step is

6

performed during camera 14 manufacturing to provide off-the-shelf web enabled cameras 14. The cameras 14 are also customized for specific entities 12 to create entity-specific cameras 14 by providing each of the cameras 14 with at least one entity ID 28 identifying the corresponding controlling entity in step 52.

Referring now to FIG. 3, a diagram showing the preferred embodiment of the connectivity and application software of the camera 14 and the entity ID 28 information is shown. Preferably, the camera 14 includes a microprocessor-based architecture that runs an operating system 70 for controlling camera hardware 72 and overall functionality of the camera 14 (e.g., taking pictures, storing pictures, and the like). An example of such an operating system 70 is the Digita™ Operating Environment developed by the assignee of the present application. The camera 14 also includes communication manager 74 software, and a TCP/IP protocol stack 76, that enables communication via the internet, as is well-known in the art. The entity ID information 28 and captured images may be stored in one or more types of memories 82.

For hierarchical entity relationships, the cameras 14 are provided with hierarchical entity IDs 28; one entity ID 28 identifying the entity, and a second entity ID 28 identifying the end-user. Whether there are one or more entity IDs 28, the entity ID 28 of the camera manufacturer may always be provided. Camera 14 customization may occur either during manufacture or anytime thereafter. In a preferred embodiment, the manufacturer entity ID 28 is provided at the time of manufacturing and is stored in an EEPROM 82b, while the entity IDs 28 for other entities 12, such as companies and end-users, are loaded into the camera 14 subsequent to manufacturing and are stored in flash memory 82a or the EEPROM 82b.

Customization that occurs subsequent to manufacture may be implemented using several methods. The first method is to manufacture the cameras 14 with an application programming interface (not shown) for accepting a subsequently loaded software application that specifies the entity ID's 28. The application may come preloaded on a flash card, which is then inserted into the camera 14 by the user and stored in flash memory 82a. The application may also be wirelessly beamed into the camera. When executed in the camera, the software application transmits the appropriate entity IDs 28 to the gateway server 18.

The second method is to load a small file in the camera 14 specifying the entity IDs 28 from a removable memory or from a PC, and storing the file in a system folder within the camera's flash memory 82a. The camera 14 then accesses the file when an Internet connection is established. In a preferred embodiment, the communication manager 74 automatically extracts the manufacturing ID 28 and the entity ID 28 and transmits them to the gateway server 18. In this embodiment, the entity ID 28 is also stored in the EEPROM 82b and is factory set to zero (empty). Thus, unless the entity ID 28 is set, the manufacturing ID 28 may default as the highest controlling entity.

If, for example, a third party developer X contracts to provide custom camera software for camera manufacturer Z, then a custom entity ID will be issued for developer X and developer X will place the custom entity ID into the EEPROM 82b. Developer X is now a controlling entity 12, and may specify to the photo-sharing service 16 that a developer X entity-specific photo-sharing site 22 or developer X's own website be the destination for the uploaded images.

7

The protocol stack 76, under direction of the communications manager interfaces with the communications hardware 78 of camera. The protocol stack 76 includes software APIs and protocol libraries that interface with the communication manager 74, and communication hardware interface drivers that interfaces directly with the various communication hardware 72 the camera 14 must function with (e.g., a Bluetooth transceiver, etc.). The communication manager 74 communicates with operating system 70 and the IP protocol stack 76 to establish an Internet connection ant to transmit the entity ID 28 information and images from the memories 82a and 82b to the photo-sharing service 16.

In an alternative embodiment, rather than loading entity ID's 28 into the camera, a combination of the camera's serial number and the make and model number of the camera may be used as the entity ID 28. Entity specific cameras 14 may then be distinguished by providing a mapping of the camera serial numbers and product IDs to specific entities 12 in the database 20.

Although the camera 14 has been described in terms of a software-based customization solution, those with ordinary skill in the art will readily recognize that the camera 14 may also be provided with a hardware-based solution.

Referring again to FIG. 2, before or after camera customization, the entity-specific websites 22 are created for each entity contracting with the photo-sharing service 16 in step 54. Customization requires storage of entity information in the entity/account database 20 and creating and storing web page elements comprising the entity-specific photo-sharing website in the database 20. The entity-specific information stored in the database 20 may also include service levels, and enabled features for the entity-specific website 22. Features are components or services that may be provided on websites by the photo-sharing service 16, such as search functions, and online printing, for instance, but may be selectable by each entity for its own website. As an example, company X may provide customized cameras 14 for its employees, but may not wish to allow employees to print images from the company X photo website for security reasons. If so desired, company X may have the photo service disable this feature from their particular website.

In a preferred embodiment, the entity-specific websites 22 are not created from scratch, but are created by modifying a preexisting template. The template may include several different sections, such as A, B, C and D, for instance. Assuming for example that the template used to create a website for Nikon, and section A is used to specify the name of the entity then the name Nikon would be inserted into that section. Other entity-specific content would be used to fill out the remaining sections. The Web pages comprising the Nikon specific photo-sharing website would then be provided with URL's unique to that website. The entity's regular website would be modified to include a link to the entity's photo-sharing website 22. In addition, the entity-specific photo-sharing website would include a link back to the entity's website. Entities 12 may have entity photo-sharing websites 22 created for them in one of two ways; automatically by logging into the photo-sharing service 16 and manually customizing the templates, or by having the entity photo-sharing website created for them.

Referring still to FIG. 2, when a camera 14 establishes an internet connection with the gateway server 18, the camera 14 transmits its entity IDs 28 and/or user entity ID 28 when uploading user selected images to the gateway server 18 in step 56. In response, the gateway server 18 matches the entity ID 28 with an entity account in the database 20 and

8

matches the user entity ID 28 with a user account 40 in the database 20 in step 58. The images received are then stored in the user account 40 of the identified entity-specific website 22 in step 60.

Referring again to FIG. 1, each user account 40 in the database 20 may also include one or more action lists 48. According to the present invention, an action list 48 includes one or more items representing actions that the gateway server 18 should take with respect to uploaded images, such as where to store and/or send the images from a particular user or camera, for instance. As explained further below, the action list 48 stored on the database 20 under a user's account 40 are automatically downloaded to the user's camera 14 during a connection with the gateway server 18 and stored on the camera 14. When the user initiates an image upload, the action list 48 is displayed to the user so the user may easily select what actions the gateway server 18 should take with respect to the images by selecting the displayed action list items.

Examples of action list items include specifying that the uploaded images should be stored on the entity-specific photo website, sending the images to a list of email addresses, or even performing some type of analysis or calculation on the image data, for instance.

In a further aspect of the present invention, an action list item is not limited to, instructing the gateway server 18 to perform actions only within the photo-sharing service 16. Rather, an item in the action list 48 may also instruct the gateway server 18 to perform actions outside of the photo-sharing service 16, such as storing the images in an external database 49 of the entity 12. For instance, in the example where the entity 12 is a company, some users of the company's cameras 14 could have action lists 48 instructing the gateway server 18 to store uploaded images to the company's database, rather than to the company's photo-sharing site 22. Based on the action lists 48 and customization, the gateway server 18 may be programmed to automatically perform predefined tasks, such as creating new web albums, or a new page within an existing album, parse the images to extract sound files or other metadata, print images and mail them to designated addresses, and so on.

In a preferred embodiment, the action lists 48 may be created via several methods. In one method, the action list is created by the photo-sharing service 16 the first time the user's camera establishes a connection. That is, a default action list 48 is automatically created based on the entity ID when a user account 40 is first created. In a hierarchical entity relationship where the entity 12 is a company, a default action list 48 may be created to implement a workflow specified by the entity 12. In a hierarchical relationship where the entity 12 is camera manufacturer, for instance, a default action list 48 may be created instructing the gateway server 18 to store the user's images in a simulated "shoebox" on the entity-specific photo-sharing site 20. The user may then go online and create albums from the images in the shoebox as desired.

Another method is for the user to create the action list 48 online on the entity-specific photo-sharing site 20. The action list 48 may be created manually on the website 20 by the user navigating to the site 20 using a web browser 24, accessing her account, and manually creating the action list 48 or editing the action list 48 on the entity-specific site 22. The action list 48 may also be created automatically on the website 20 in response to user actions performed on the website, such as printing images, or creating a web album.

9

Alternatively, after performing an action, the user may be prompted whether they would like this action added to his or her action list 48. If so, the user clicks a check-box and the item is added the action list 48. In a preferred embodiment, any action list 48 created and edited on the photo-sharing site 20 are downloaded to the camera every time the camera 14 connects to the photo-sharing service 16 and made available for user selection on the camera 14 during the next upload.

Yet another method for creating an action list 48 is to allow the user to create the action list on the camera 14. The user may manually create an action list 48 by "typing" in predefined items on the camera. The user may also type in an email address as an action list item whereby when that item is selected, the uploaded images are stored as a web album on the entity-specific photo-sharing website 22 and the server 18 sends a notification to the specified recipient containing the URL to the web album page.

A method and system for hosting web-based photo-sharing websites and for customizing digital cameras to upload images to the entity-specific photo-sharing websites has been disclosed. According to the present invention, users of the customized cameras 14 can upload images to the Internet for storage and web photo album creation without the use of a PC.

In one embodiment described above, the present invention assumes that an ISP account has been established with the digital camera's service provider, and that users of cameras 14 belonging to a certain entity 12 may use the cameras 14 to upload images to the website of the entity 12. However, two problems with account setup remain. One problem is that just as with a PC and PDA, the user must first establish an ISP account before the camera 14 can establish Internet communication. The second account problem is that most websites, including the photo sharing sites 22, may require each user to establish a unique account before using the site to distinguish one user from another. Before being able to connect the web-enabled camera 14 to the Internet, the user must establish these two accounts by either entering account setup information on a PC or entering account setup information on the camera 14. Neither alternative is a convenient alternative for people who do not have the time nor inclination to do so.

In a further aspect of the present invention, the cameras and the photo sharing site are provided with software for automatically creating Internet and photo-sharing website accounts for each camera 14 upon first use, without requiring the user to first enter account information on a PC or on the camera 14.

Referring now to FIGS. 4A and 4B, a flow chart illustrating a process for automatically configuring a web-enabled digital camera to access the Internet in a preferred embodiment of the present invention. Although the process will be described in terms of automatically establishing Internet accounts for a digital camera without requiring the user to enter information, those with ordinary skill in the art will readily recognize that the present invention may be used to automatically establish Internet accounts for any type of portable electronic device.

The process assumes that a user has just acquired a digital camera 14 customized as described above, and has just taken the camera 14 out of its box in step 100. After taking pictures with the camera in step 102, the user may review the images in the camera's LCD screen and optionally select a set of images to upload to the photo sharing service 16 in step 104. The user then presses a "send" button on the camera in step 106 to upload the images.

10

In response, the camera displays a confirmation dialog screen on the camera and prompts the user to record an audio annotation for the images or to continue in step 108. The user may then choose to record audio for the images in step 110. After choosing to continue, or after recording audio, the camera displays a "connecting" dialog screen in step 112 to indicate to the user that the camera is establishing an Internet connection. At the same time, the camera checks for available connection options in step 114, and if more than one is found, the camera prompts the user to select one of the connection options. For example, the camera may be within range of a Bluetooth-equipped printer and a cellphone, so the user will be prompted to choose which device the camera should establish communication with.

The camera then checks for the presence of account information on the camera in step 116, and if there are none, the camera assumes that this is a first-time connection. According to the present invention, in order to allow the camera to make a first-time Internet connection, the camera is provided with default Internet service provider (ISP) information during manufacturing, including an ISP access number, and user ID and password (if required). The camera establishes connection with the default ISP in step 118 by dialing the preloaded access number, and by sending the preloaded user ID and password to the ISP. This special account may be configured so that the camera can only connect to the gateway server 18 (no other IP addresses may be allowed).

After connecting with the ISP, the camera connects to the gateway server 18 and sends unique camera information and/or user information in step 120. In a preferred embodiment of the present invention, a combination of the camera's serial number and the make and model number of the camera may be sent as the unique camera information. In another preferred embodiment, the user's e-mail address may be sent as the unique camera or user information.

Continuing with FIG. 4B, the gateway server 18 uses the unique camera information to set up a user account 40 in step 122. After creating the user account 40, the gateway server 18 returns an information packet to the camera containing new ISP information (if needed), an account ID, and an account password in step 124. The information packet may also contain a default action list specifying what actions should be taken with respect to the images, an advertisement for display on the camera, and the URL of the entity-specific website 22.

It should be noted that if the camera is used in conjunction with an IP direct phone or is provided with a phone number for connected to a dedicated server where the user is not billed separately for the ISP connection, then the steps of providing the camera with default ISP info and returning new ISP info, may be omitted.

The camera receives the information packet in step 126, and writes a configuration file to memory 82a containing the ID, password, and default action list in step 128. The camera then displays the action list on the camera's LCD screen for selection by the user in step 130. The camera may optionally display the user's account information as well.

In an alternative preferred embodiment where security is a concern, the camera 14 first logs off the special ISP and the gateway accounts, and then reconnects using new ISP and gateway accounts in order to retrieve the action lists 48.

FIG. 5 is a block diagram illustrating an example action list 48 displayed on the LCD screen of the camera 14. The action list 48 is shown displaying three major options; printing the uploaded images, saving the uploaded images in

11

the user's shoebox, and sending the images to Mom. Under the printing option, the user may select from various size prints. Rather than nested menu categories as shown under the printing option, the action list 48 may be displayed with each action listed as a separate item (e.g., "Send 4x8 prints to Mom", "Send 5x7 prints to me").

Referring again to FIG. 4B, after the user selects one or more actions from the action list 48, the camera begins to upload the images along with the selected actions in step 132 and displays a progress bar on the screen in step 134. In one preferred embodiment, the camera may also display the advertisement sent in the information packet from the gateway server 18. The advertisement may advertise the controlling entity 12, the entity's photo-sharing site 22, or the photo sharing service 16. After all the images are uploaded and associated with the user's account 40, the camera breaks the connection with the gateway server in step 136. At this point, the camera 14 may also display the URL of the entity-specific website 22 to the user.

The next time the user uploads images, the camera will use any new ISP information received to connect to the Internet and will use the account ID and password written to memory 82a when connecting to the gateway server 18. Thus, by using unique camera information, such as the serial number, to establish a web site account upon first use, the present invention eliminates the need for the user to type in information to establish a web site accounts.

To further explain the present invention from a user interaction point of view, consider the following scenario where a user named Jack has just purchased a digital camera 14 from a store and unpacks the camera from the box. There is a "Quick Start Guide" which guides him in getting started. Jack pops in the batteries, sets the date and time, and takes some pictures of his dog and his new baby. Jack can see the pictures have come out well on the small LCD, and now wants to try sharing them with his parents.

The Quick Start Guide says to select the photos to be sent using the "Select" button, and then press the "Send" button. So, Jack navigates to each of the baby pictures, selects them, and then presses the "Send" button. Instantly, a dialog comes up on the LCD screen: "No Receiving Device Found! Please turn on your phone or other connecting device." Oops! Jack pulls out his cell phone, and turns it on. He presses the "Continue" button. Jack does not see this happen, but the camera now "discovers" the cell phone, and immediately presents another dialog: "4 Images Selected. Press Record to add a Sound Note, or Continue to send". Although Jack finds the proposition of recording sound intriguing, he decides to skip it and presses the "Continue" button. Immediately, the dialog is replaced with a "Connecting . . ." dialog.

Shortly, another dialog appears: "Your camera serial number is 38147. Please write this down. You will need it to access your web photo albums". Jack writes the number down on the spot provided in the Quick Start Guide, and presses OK.

Another "Connecting . . ." dialog appears, and is then quickly replaced by another dialog, which says "A free, temporary account has been set up for you at www.photo-sharing.service.com/new_accounts. You will need your camera serial number to access your photos and complete the setup of your account. Please complete the account setup within 30 days". Jack writes down the URL in the space provided in the Quick Start Guide, and presses the OK button. Jack doesn't know it, but during this dialog, the camera has begun transmitting his images and is already partially complete.

12

A new dialog comes up, with a progress bar. Jack is surprised to see that the transmission is already almost ½ done. Below the progress bar, the dialog says "Press 'Continue' to use camera during photo transmission, or wait for progress bar to complete". Jack is interested in watching how fast his images are transmitted, and decides to watch the progress bar complete. A "Transmission Successful" dialog appears. Jack presses the OK button. The camera returns to the review mode.

Jack is pretty excited—he just sent four baby pictures to the Internet. Jack then decides to see what happened to his images so he turns on his PC. After connecting to the Internet, Jack types in the URL from the Quick Start Guide. A Photo-sharing service web page appears, welcoming Jack to the photo-sharing service. After looking briefly at the welcome page, Jack types in his serial number from the Quick Start Guide, and selects his camera's model number from a pop-up menu. Jack clicks on a "Submit" button on the web page.

Jack now sees a page which shows thumbnails of the baby pictures he just sent, the page is entitled "My Shoebox". The page explains to Jack that he is looking at his on-line digital photo shoebox. Since the server 18 knows this is Jack's first visit, special help messages may appear. Various options are provided via buttons and text links. One that catches Jack's eye is CREATE WEB PHOTO ALBUM. Jack clicks this button, and works his way through the process of setting up an on-line photo album. This includes selecting photos from the shoebox, as well as selecting layout and style. One of the check-box items Jack is offered is "Make this album a camera Action List Item". Jack doesn't know what that is, so he clicks the Action List link, which brings up a brief description "If you check this box, you will be able to send pictures directly from the camera to this photo album!" Jack finds this interesting, so he closes the description window, and checks the box. Jack also enters the email address for his parents and his wife's parents, so they can be notified to come and see his photo album, which he has entitled "Our First Baby".

One of the buttons Jack does not click is the "Complete Account Setup" button. He knows that he has 30 days to do that chore, and figures he will get back to it later.

Jack's wife arrives home from shopping at this point, and Jack wants to show her how the new camera works. He starts by showing her the baby album on the PC, then decides to take some pictures of them holding the baby. Jack then selects the images, and presses the "Send" button again.

Since his cell phone is still on, sitting on the table a few feet away, the connection goes smoothly and quickly. A new dialog pops up, surprising Jack. It says "Select the destination for your pictures" and offers two choices: "My Shoebox" and "Our First Baby". Jack is amazed, he doesn't realize that the server 18 downloaded his action list to his camera during the connection. Jack decides to select the "Our First Baby" web album as the destination for the images, and clicks OK. The pictures are sent as before. After the transmission has completed, Jack goes to the PC to check on the photo album. When Jack refreshes the album page, he now sees the additional pictures he just sent, along with the pictures he sent before.

Jack spots a "Send Prints" link on the web page, and clicks it. He is led through a selection of print types, mailing addresses, and credit card info to make it possible to send prints. He is offered the option of completing the setup of his account. Jack decides to do that now, and proceeds to fill out the requested information, including his credit card number.

13

Once the account setup is complete, Jack continues the print order. One of the radio button items is "Make prints a separate Action List Item" or "Make prints part of your Action List". Jack remembers something about Action Lists from before, but is not sure what this means. The description says "Making a separate action list for prints allows you to decide in the camera to send to the photo album and send prints or to just send to the photo album." Jack thinks this is cool, and clicks the "Make prints a separate Action List" item. The next time Jack sends photos from the camera, his action list will be updated to present three choices: My Shoebox, Our First Baby, and Our First Baby w/Prints.

The underlying technology supporting this scenario are summarized below. Other functions and features are assumed, but not required for this scenario:

1. Two-way connection between camera and portal
2. Camera metadata included in the request
3. If not using an IP direct connection,
 - 3.1. Default ISP connection info built into the camera for the first connection (country specific)
 - 3.2. Downloaded assigned ISP information from the portal to the camera
4. Software capable of recognizing automatically a set of supported phones and adjusting the protocol to match
5. Action Lists maintained on the server that are automatically downloaded to the camera to update the camera selection list each time a connection is made
6. An On-line shoebox
7. Ability to download files to the camera from the server. The files could be text, GIF, animated GIF, JPG, or even a script or applet. This feature enables the ability to display advertisements on the camera, remind the user of remaining time to complete account setup, make special offers, and indicate limits reached.

A method and system for automatically configuring a web-enabled digital camera to access the Internet has been disclosed. Although the present invention has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations to the embodiments and those variations would be within the spirit and scope of the present invention. For example, although the photo-sharing service has been described as including the gateway server and the database, the database may be located elsewhere. Also, the gateway server may be used to control account information, while one or more other servers may be used to provide the web pages of the entity-specific websites. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.

What is claimed is:

1. A method for automatically establishing a user account and for configuring a hand-held image capture device which includes a digital camera for accessing a site on a public network using the user account, the method comprising the steps of:

- (a) establishing a connection to a website server, the website server being responsible for establishing and maintaining website accounts;
- (b) checking for the presence of account information on the image Capture Device and if none are found, determining that this is a first-time connection and automatically sending image capture device information uniquely identifying the electronic device to the website server so that the server can set-up the account information based on the image capture device information;

14

(c) receiving user account information from the server, wherein the user account information includes an account ID and password; and

(d) storing the user account information on image capture device for use a next time the image capture device accesses the website, whereby the user does not have to enter account information in order to establish the connection with the public network or to establish the website account.

2. The method of claim 1 further including the step of providing a digital camera as the hand-held image capture device.

3. The method of claim 2 further including the step of sending at least a partial serial number as the image capture device information uniquely identifying the electronic device.

4. The method of claim 3 further including the step of uploading captured images to the website server.

5. The method of claim 4 wherein step (a) further including the steps of:

- (i) providing the image capture device with default ISP information; and
- (ii) establishing a connection to an ISP using the default ISP information.

6. The method of claim 5 further including the step of terminating the connection with the ISP when the uploading of the images is complete.

7. The method of claim 6 further including the step of providing the digital camera with a default ISP access number, user ID, and password.

8. A system for automatically establishing a user account and for configuring a hand-held image capture device which includes a digital camera image capture device for accessing a site on a public network using the user account, comprising:

means for establishing a connection to a website server, the website server being responsible for establishing and maintaining website accounts;

means for checking for the presence of account information on the image capture device, and if none are found, determining that this is a first-time connection and automatically sending image capture device information uniquely identifying the image capture device to the website server so that the server can set-up the account information based on the image capture device information; and

means for receiving user account information from the server, wherein the user account information includes an account ID; and

means for storing the user account information on image capture device for use a next time the image capture device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account.

9. The system of claim 8 wherein the hand-held image capture device comprises a digital camera.

10. The system of claim 9 wherein the image capture device information uniquely identifying the image capture device includes a serial number of the image capture device.

11. The system of claim 10 wherein the camera uploads captured images to the website server.

12. The system of claim 9 wherein the camera further includes means for providing the image capture device with default ISP information, and means for establishing a connection to an ISP using the default ISP information.

13. The system of claim 12 wherein the digital camera terminates the connection with the ISP upon completion of image uploading.

15

14. The system of claim 13 wherein the camera is provided with a default ISP access number, user ID, and password.

15. A method for automatically establishing a user account and for configuring a digital camera to access a website on a public network using the user account, the digital camera including captured images and communication means for accessing the public network, the method comprising the steps of:

- (e) storing default ISP information on the digital camera, the default ISP information including an access number;
- (f) in response to a user request to upload images to the website, determining whether this is a first-time connection;
- (g) if it is first-time connection, establishing communication with the default ISP by dialing the access number;
- (h) establishing communication with the website and automatically sending a digital camera ID to the website;
- (i) using the digital camera ID to set up a user account on the website;
- (j) sending user account information to the digital camera, the user account including an account ID and an account password;
- (k) storing the user account information on the camera; and
- (l) uploading images to the website and associating the images with the user account, whereby the user does not have to enter information to establish a connection with the public network or to establish the website account.

16. The method of claim 15 further including the step of sending at least a partial serial number as the digital camera ID.

17. The method of claim 16 further including the step of receiving new ISP information from the website.

18. The method of claim 17 further including the step of using the new ISP information a next time the digital camera establishes a connection with the public network.

16

19. A system for automatically establishing a user account and for configuring a digital camera for accessing a website on a public network using the user account, the digital camera including images captured by user and communication means for wirelessly accessing the public network, the system comprising the steps of:

- means for storing default ISP information on the digital camera, the default ISP information including an access number;
- means for determining whether this is a first-time connection in response to a user request to upload images to the website;
- means for establishing communication with the default ISP by dialing the access number;
- means for establishing communication with the website and sending a digital camera ID to the website;
- means for using the digital camera ID to set up a user account on the website;
- means for automatically sending user account information to the digital camera, the user account including an account ID and an account password;
- means for storing the user account information on the camera; and
- means for uploading images to the website and associating the images with the user account, whereby the user does not have to enter information to establish a connection with the public network or to establish the website account.

20. The system of claim 19 wherein the digital camera ID includes a serial number.

21. The system of claim 20 wherein the digital camera receives new ISP information from the website.

22. The system of claim 21 wherein the digital camera uses the new ISP information a next time the digital camera establishes a connection with the public network.

* * * * *



US00635384B1

(12) **United States Patent**
Morris

(10) **Patent No.:** **US 6,353,848 B1**
(45) **Date of Patent:** **Mar. 5, 2002**

(54) **METHOD AND SYSTEM ALLOWING A CLIENT COMPUTER TO ACCESS A PORTABLE DIGITAL IMAGE CAPTURE UNIT OVER A NETWORK**

(75) Inventor: **Robert P. Morris**, Raleigh, NC (US)

(73) Assignee: **FlashPoint Technology, Inc.**,
Peterborough, NH (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/127,514**

(22) Filed: **Jul. 31, 1998**

(51) Int. Cl.⁷ **G06F 15/16**

(52) U.S. Cl. **709/203; 709/200; 709/202; 709/217; 709/219; 709/238; 709/232; 709/245**

(58) **Field of Search** **709/200-203, 709/206, 217-219, 227-229, 231-232, 236-237, 245; 707/10, 200-204; 713/201-202**

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,911,044 A	~	6/1999	Lo et al.	709/203
6,018,774 A	~	1/2000	Mayle et al.	709/203
6,058,428 A	~	5/2000	Wang et al.	709/232
6,085,249 A	~	7/2000	Wang et al.	709/229
6,101,536 A	~	8/2000	Kolani et al.	709/217
6,141,759 A	~	10/2000	Braddy	709/203

FOREIGN PATENT DOCUMENTS

DE 198 08 616 9/1998 H04L/12/16

OTHER PUBLICATIONS

De Albuquerque et al., "Remote Monitoring Over The Internet", Nuclear Instruments & Methods In Physics Research, Section-A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 412, No. 1, Jul. 21, 1998, pp. 140-145.

* cited by examiner

Primary Examiner—Zarni Maung

Assistant Examiner—Bharat Barot

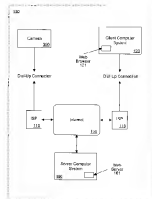
(74) Attorney, Agent, or Firm—Sawyer Law Group LLP

(57)

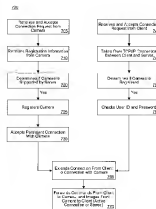
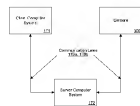
ABSTRACT

A method for accessing a digital image capture unit via a communication network comprising a server computer system and a client computer system communicatively coupled with communication equipment. In one embodiment, the address of the digital image capture unit is registered in an executable program on the server computer system. The executable program is accessed by the client computer system. The executable program connects the digital image capture unit and the server computer system. The executable program communicates commands between the client computer system and the digital image capture unit, such that data captured by the digital image capture unit is transferred to the client computer system via the server computer system.

27 Claims, 11 Drawing Sheets



126



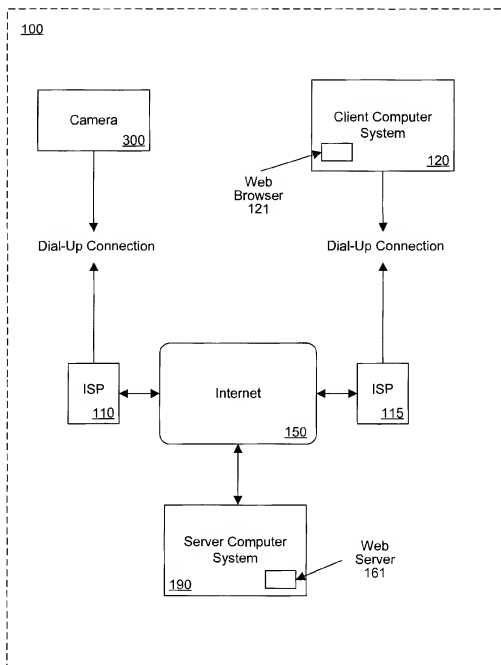


FIG. 1A

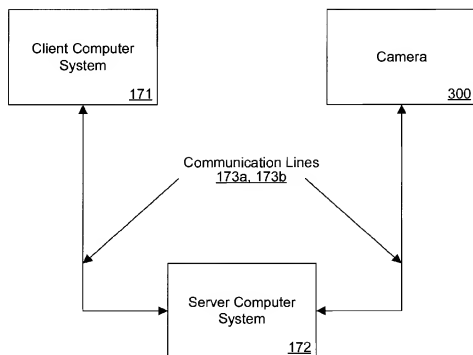
170

FIG. 1B

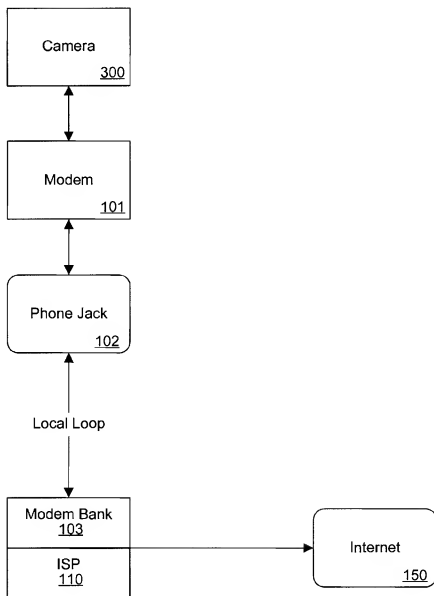


FIG. 1C

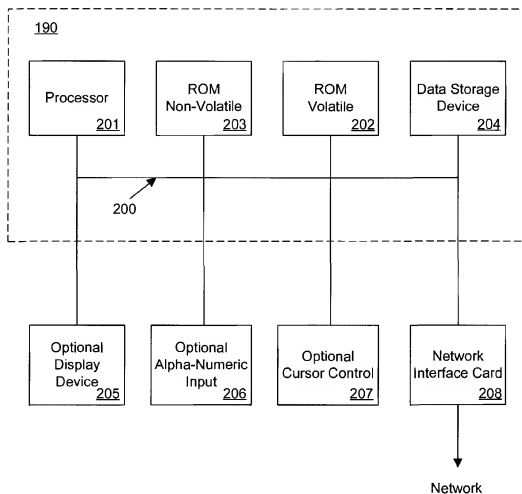


FIG. 2

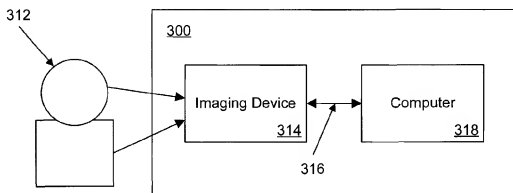
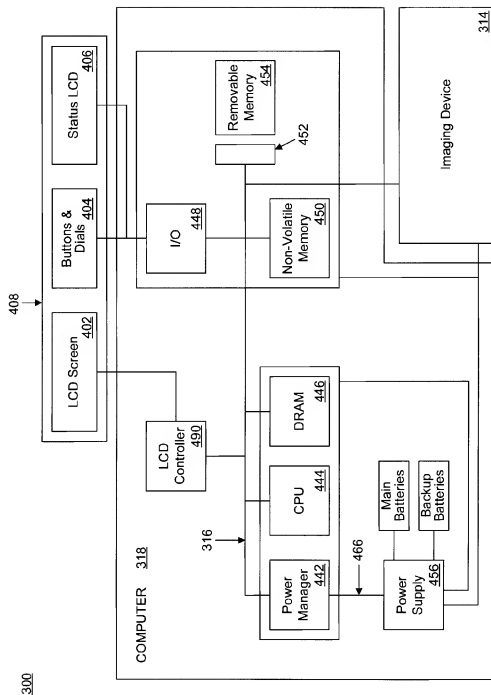


FIG. 3



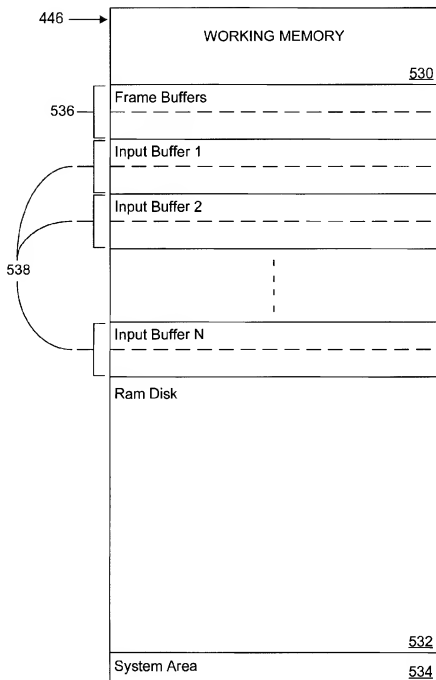


FIG. 5

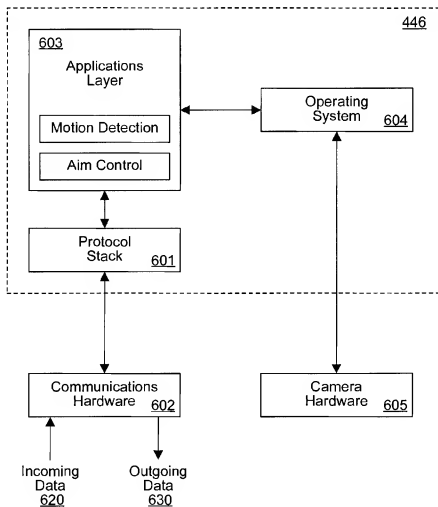


FIG. 6

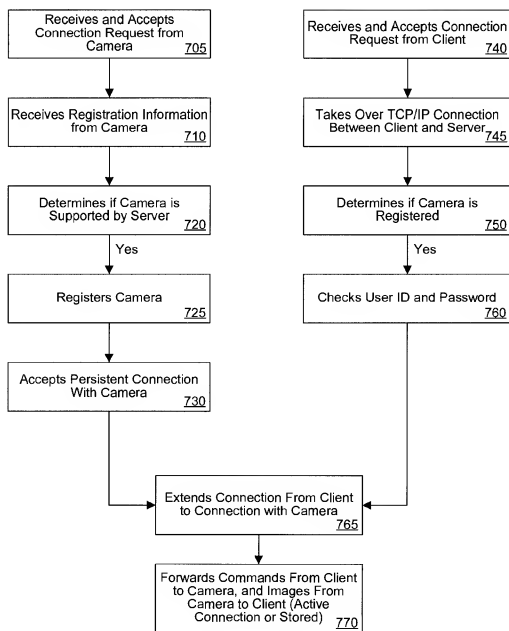
700

FIG. 7

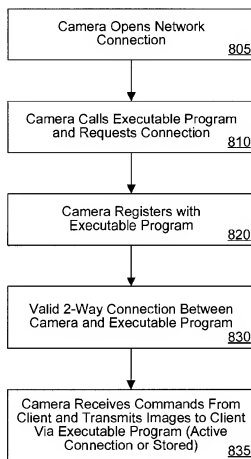
800

FIG. 8

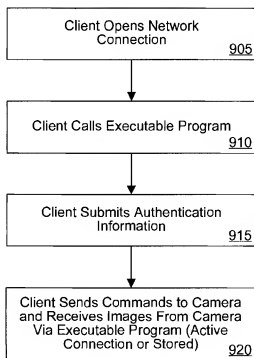
900

FIG. 9

METHOD AND SYSTEM ALLOWING A CLIENT COMPUTER TO ACCESS A PORTABLE DIGITAL IMAGE CAPTURE UNIT OVER A NETWORK

FIELD OF THE INVENTION

The field of the present invention pertains to digital image capture devices. More particularly, the present invention relates to a method for remotely accessing a digital camera via a communication network.

BACKGROUND OF THE INVENTION

Modern digital cameras typically include an imaging device which is controlled by a computer system running a software program. When an image is captured, the imaging device is exposed to light and generates raw image data representing the image. The raw image data are typically stored in an image buffer, where they are processed and compressed by the computer system's processor. Many types of compression schemes can be used to compress the image data, such as the joint photographic expert group (JPEG) standard. After the processor processes and compresses the raw image data into image files, the processor stores the image files in an internal memory or on an external memory card.

Some digital cameras are also equipped with a liquid-crystal display (LCD) or another type of display screen on the back of the camera. Through the use of the LCD, the processor can cause the digital camera to operate in one of two modes, play and record, although some cameras only have a record mode. In the play mode, the LCD is used as a playback screen allowing the user to review previously captured images either individually or in arrays of four, nine, or sixteen images. In the record mode, the LCD is used as a viewfinder through which the user may view an object or scene before taking a picture.

Besides the LCD, user interfaces for digital cameras also include a number of buttons or switches for setting the camera into one of the two modes and for navigating between images in play mode. For example, most digital cameras include two buttons, labeled "-", and "+", that enable a user to navigate or scroll through captured images. For example, if the user is reviewing images individually, then pressing one of navigation buttons causes the currently displayed image to be replaced by the next image.

A digital camera has no film and, as such, there is no incremental cost of taking and storing pictures. Hence, it is possible to take an unlimited number of pictures, wherein the most recent picture replaces the earliest picture, for virtually zero incremental cost. Accordingly, this advantage is best realized when the camera is used as much as possible, taking pictures of practically anything of interest.

One way to best utilize this unique attribute is to make the digital camera and its internally stored images remotely accessible. If the pictures are remotely accessible, the camera could be set to continuously take pictures of scenes and items of interest. Ideally, a user would be able to access those pictures at any time. The user would be able to use a widely available communications medium to access the camera from virtually an unlimited number of locations.

The emergence of the Internet as a distributed, widely accessible communications medium provides a convenient avenue for implementing remote accessibility. Providing remote accessibility via the Internet leverages the fact that the Internet is becoming familiar to an increasing number of

people. Many users have become accustomed to retrieving information from remotely located systems via the Internet. There are many and varied applications which presently use the Internet to provide remote access or remote connectivity. Internet telephony is one such application, such as Microsoft's NetMeeting and Netscape's CoolTalk.

NetMeeting and CoolTalk are both real-time desktop audio conferencing and data collaboration software applications specifically designed to use the Internet as their communications medium. Both software applications allow a "local" user to place a "call" to a "remote" user located anywhere in the world. With both NetMeeting and CoolTalk, the software application is hosted on a personal computer system at the user's location and on a personal computer system at the remote user's location. Both NetMeeting and CoolTalk require a SLIP (Serial Line Internet Protocol) or PPP (Point-to-Point Protocol) account where Internet access is via a dial-up modem, and where the user, as is typical, accesses the Internet through an ISP (Internet Service Provider). Both NetMeeting and CoolTalk require personal computer systems for the resources necessary to run these applications (e.g., processing power, memory, communications hardware, and the like). In addition, both NetMeeting and CoolTalk require the one user to input an IP (Internet Protocol) address for the other user in order to establish communication between the users.

To facilitate the process of obtaining appropriate Internet addresses, CoolTalk, for example, allows on-line users to list their respective IP addresses with a proprietary CoolTalk central Web server. This allows a user to obtain a list of users currently on-line to whom communication can be established. Upon locating the desired remote user in the Internet address list maintained by the Web server, the local user places the call.

In this manner, the proprietary CoolTalk Web server maintains a user-viewable and user-updated "address book" in which users list their respective Internet addresses and in which they search for the Internet addresses of others with whom they wish to communicate. However, both NetMeeting and CoolTalk require active user input, in that each require the user to input his current Internet address, and in that each require the local user to search the address book for the Internet address of the remote user to be contacted. This can be quite problematic in the case where users obtain access to the Internet via dial-up connections and hence have different Internet addresses each time their respective dial-up connections are established.

In a manner similar to Internet telephony, Internet desktop video conferencing is another prior art application which uses the Internet as its communications medium. One such application, for example, is CU-SeeMe by White Pine. CU-SeeMe provides real time video conferencing between two or more users. As with NetMeeting and CoolTalk, CU-SeeMe is a software application which runs on both the local user's personal computer system and the remote user's personal computer system. The personal computer systems provide the resources for running the application. As with NetMeeting and CoolTalk, CU-SeeMe requires the local user to enter the IP address of the remote user. Like CoolTalk, CU-SeeMe facilitates this process by allowing on-line users to list their respective IP addresses with a proprietary central Web server such that the addresses can be easily indexed and searched.

Another prior art example of remote access via the Internet is status queries of remote devices using the Internet as the communications medium. A typical prior art applica-

3

tion involves interfacing a remote device with a computer system, and accessing the computer system via the Internet. For example, a vending machine can be remotely accessed to determine its status (e.g., the number of sales made, whether the machine needs refills, whether the machine needs maintenance, and the like). The machine is appropriately equipped with sensors, switches, and the like, which in turn are interfaced to a computer system using a software driver. The computer system is coupled to the Internet and interfaces with the machine through the driver, making the relevant information available over the Internet using Web server software. Hence, any interested user (e.g., the vending machine service company) is able to remotely ascertain the status of the machine via the Internet.

A problem with the above described prior art applications is that access to the Internet and communication thereon require a separate host computer system (e.g., a personal computer system) on each side of the Internet connection in addition to the server computer system on the Internet. The two host computer systems provide the computational resources to host the respective software applications, the Internet access software, and any necessary device drivers. The required computational resources consume a significant amount of memory. Because of this, among other reasons, the above prior art applications are not easily transferred to the realm of easy-to-use, intuitive, consumer electronic devices such as digital cameras, which are small in size and so generally constrained by the amount of memory they can house. In addition, a consumer electronic device such as a digital camera that requires a separate computer system would be more expensive and complex, and therefore would not be consistent with the desire of consumers for lower cost and simpler devices.

Also, separate host computer systems (where the host computer systems host the software and drivers required by prior art applications as described above) require extra effort to administer, particularly with regard to networks consisting of a large number of computer systems (e.g., digital cameras each incorporating a computer system). For example, an upgrade to the software residing on each computer system has to be individually installed on each computer system. Also, each computer system has to be individually polled to query whether the computer system has data of interest to the user, and then the data have to be separately accessed and collected from each computer system, then compiled. For example, in an application involving digital cameras, a user may be interested in finding out which digital cameras have images in storage. In the prior art, the user has to access each digital camera individually. In another case, a user may have an interest in maintaining a record of transactions between all users and all digital cameras. Again, in the prior art this is accomplished by individually accessing each digital camera (or, alternatively, each user's computer system) to collect the data, and then compiling the complete list of transactions.

Another problem with the prior art is the fact that the applications described above require the user to know the Internet address of the person or device that is being contacted. The Internet telephony applications (e.g., CoolTalk) often employ a user-viewable and user-updated address book to facilitate the process of locating and obtaining the correct Internet address; however, they require active user input. This is difficult in the case where users obtain access to the Internet via dial-up connections, and thus have changing Internet addresses. Still another problem with the prior art is that the applications described above provide only a limited degree of functionality; that is, they are

4

limited to either chat, video conferencing, or the like. As such, they are not capable of establishing a connection between any type of user system and remote device.

One prior art system is described in the copending previously filed patent application, assigned to the assignee of the present invention, entitled "A Method and System for Hosting an Internet Web Site on a Digital Camera," Eric C. Anderson and others, Ser. No. 09/044,644. This prior art system presents one solution to the problem of gaining remote access to those digital devices where the location and Internet address of the device are highly changeable. This prior art system incorporates a Web server into the digital device, specifically a digital camera. However, the disadvantage to this prior art system is that the Web server consumes valuable memory and computational resources in the digital camera. In addition, because of the limited memory in a device such as a digital camera, the Web server is not as powerful as a Web server on a server computer system.

Thus, a need exists for an inexpensive and powerful method for implementing remote access to digital devices, such as digital cameras, where the location and Internet address of the device are highly changeable. A further need exists for an intuitive, simple protocol for presenting the device's functionality and capabilities to users. In addition, a need exists for a method of efficiently administering a plurality of separate devices. A need also exists for an efficient process of obtaining the address of the device that is transparent to the viewer. The present invention provides a novel solution to the above needs.

SUMMARY OF THE INVENTION

The present invention provides an inexpensive and powerful method for implementing remote access to digital devices, such as digital cameras, where the location and Internet address of the device are highly changeable. The present invention further provides an intuitive, simple protocol for presenting the device's functionality and capabilities to users. In addition, the present invention provides a method of efficiently administering a plurality of separate devices. The present invention also provides an efficient process of obtaining the address of a device that is transparent to the user.

In one embodiment, the present invention is an executable program for accessing a digital camera via a communication network using a Web server on a server computer system and a Web browser (or a program of similar function) on a client computer system that are communicatively coupled via the Internet. The address of the digital camera is registered in an executable program on the server computer system. The executable program is accessed by the client computer system. The executable program connects the digital camera and the server computer system. The executable program enables the client computer system and the digital camera to communicate using any protocol used by these devices, thus allowing data (e.g., images) acquired by the digital camera to be transferred to the client computer system.

The executable program can be implemented in a variety of forms. For example, the executable program can be a Java script. Alternatively, the executable program can be a cgi-bin (Common Gateway Interface-binaries).

For example, in the case of a digital camera, the executable program directly communicates commands from the client computer system to the digital camera when both the client computer system and the digital camera are on-line at the same time. Alternatively, if the digital camera is not

5

on-line, the commands from the client computer system are first stored in the server computer system, and then later communicated by the executable program to the digital camera after a connection between the server computer system and the digital camera is made. The capability to store and then forward commands and data is not limited to a digital camera application nor is it limited to a particular data storage format. The data storage format can be any format that is understood by both the client computer system and the digital device.

Images and any other data acquired by the digital camera are accessed by the server computer system using the executable program and directly transferred to the client computer system when both the client computer system and the digital camera are on-line at the same time. Alternatively, if the client computer system is not on-line, the data are first stored by the server computer system, and then later communicated to the client computer system after a connection between the server computer system and the client computer system is made.

It should be noted, however, that the present invention can be readily modified to function in other embodiments, such as, for example, hand-held digital devices, lap top personal computers, and the like, which require an efficient process of obtaining the address of a device that is transparent to the user.

These and other objects and advantages of the present invention will become obvious to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments which are illustrated in the various drawing figures.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not by way of limitation, in the figures of the accompanying drawings in which like reference numerals refer to similar elements and in which:

FIG. 1A shows a block diagram of a remote access system via the Internet in accordance with one embodiment of the present invention.

FIG. 1B shows a block diagram of a remote access system via a Local Area Network in accordance with one embodiment of the present invention.

FIG. 1C shows a block diagram of a digital camera coupled to the Internet via an Internet Service Provider.

FIG. 2 shows a general purpose computer system upon which embodiments of the present invention may be practiced.

FIG. 3 shows a block diagram of a digital camera for use in accordance with the present invention.

FIG. 4 shows a block diagram of a computer system of a digital camera in accordance with one preferred embodiment of the present invention.

FIG. 5 shows a memory map of a dynamic random access memory of a digital camera in accordance with one embodiment of the present invention.

FIG. 6 shows a diagram of the connectivity and application software of a digital camera in accordance with one embodiment of the present invention.

FIG. 7 is a flowchart of a process for remotely accessing a digital camera implemented by an executable program in accordance with one embodiment of the present invention.

FIG. 8 is a flowchart of a process employed by a digital camera for remote access in accordance with one embodiment of the present invention.

6

FIG. 9 is a flowchart of a process employed by a client computer system for remote access of a digital camera in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In the following detailed description of the present invention, numerous specific details are set forth in order to enable one of ordinary skill in the art to make and use the invention, and are provided in the context of a patent application and its requirements. Although the present invention will be described in the context of a digital camera, various modifications to the preferred embodiment will be readily apparent to those skilled in the art, and the generic principles herein may be applied to other embodiments. That is, any digital device which displays data, images, icons and/or other items, could incorporate the features described below and that device would be within the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

Some portions of the detailed descriptions which follow are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computer memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is here, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computer system. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, bytes, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that throughout the present invention, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes (e.g., the processes of FIGS. 7, 8 and 9) of a computer system or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present invention provides a method for making a digital device (e.g., a digital camera) and its internally stored data remotely accessible over a communication network such as the Internet or a Local Area Network (LAN). The present invention is an executable program placed on a server computer system (specifically, a Web server) that implements commands initiated by a client (or user) using a client computer system with a Web browser or a program of similar function. The present invention enables the digital camera to be set to continuously take pictures of scenes/

7

items of interest and allow a client to access those pictures at any time. The present invention allows the client to access the digital camera from virtually an unlimited number of locations and with the camera in virtually any location.

Referring now to FIG. 1A, a block diagram of communication network 100 is shown. Communication network 100 provides a method in accordance with one embodiment of the present invention which implements remote access to camera 300 and its internally stored data. Communication network 100 includes camera 300, Internet Service Provider (ISP) 110, Internet Service Provider 115, client (or user) computer system 120, and server computer system 190. ISP 110 and ISP 115 are both directly coupled to the Internet 150. Client computer system 120 includes Web browser 121 or a program of similar function, and server computer system 190 includes Web server 161. Web browser 121 interprets HTML (HyperText Mark-up Language) documents and other data retrieved by Web server 161.

In the present embodiment of the present invention, an executable program resides on server computer system 190, specifically on Web server 161. The executable program implements and manages the connection between server computer system 190, client computer system 120, and camera 300. The executable program can be implemented as a Java servlet, as a cgi-bin (Common Gateway Interface-binaries), or as a similar type of application.

With reference still to FIG. 1A, client computer system 120 is communicatively coupled to ISP 115 via a POTS (plain old telephone system) dial-up connection. Client computer system 120 is coupled to the Internet 150 via one of a bank of modems maintained on the premises of ISP 115. ISP 115 is coupled directly to the Internet via an all-digital connection (e.g., a T1 line). However, other means of coupling client computer system 120 to the Internet 150 may be used in accordance with the present invention.

As depicted in FIG. 1A, camera 300 is communicatively coupled to server computer system 190 via the Internet 150 using a dial-up connection to ISP 110 via a POTS line. Digital camera 300 accesses ISP 110 using a modem, coupling to one of a bank of modems maintained on the premises of ISP 110. ISP 110 is in turn coupled directly to the Internet 150 via an all-digital connection. However, other means of coupling camera 300 to the Internet 150 may be used in accordance with the present invention.

With reference still to FIG. 1A, it should be further appreciated that while communication network 100 shows camera 300 coupling to Internet 150 via one ISP (e.g., ISP 110) and user 120 coupling to Internet 150 via a separate ISP (e.g., ISP 115), user 120 and camera 300 could be coupled to Internet 150 through a single ISP. In such a case, user 120 and camera 300 would be coupled to two separate access ports (e.g., two separate modems out of a bank of modems) of the same ISP.

With reference now to FIG. 1B, in another embodiment of the present invention, the communication network is comprised of Local Area Network (LAN) 170. For example, LAN 170 may be a communication network located within a firewall of an organization or corporation. Client (or user) computer system 171 and server computer system 172 are communicatively coupled via communication line 173a. Client computer system 171 includes an application that is analogous to a Web browser for interpreting HTML documents and other data. Similarly, server computer system 172 includes an application analogous to a Web server for retrieving HTML documents and other data. Camera 300 can be coupled to server computer system 172 through any

8

of a variety of means known in the art. For example, camera 300 can be coupled to server computer system 172 via communication line 173b of LAN 170. This coupling can be accomplished over any network protocol that supports a persistent network connection, such as TCP (Transmission Control Protocol), NetBIOS, IPX (Internet Packet Exchange), and LU6.2, and link layers protocols such as Ethernet, token ring, and ATM (Asynchronous Transfer Mode). Alternatively, camera 300 can be coupled to server computer system 172 via an input/output port (e.g., a serial port) of server computer system 172.

Referring now to FIG. 1C, a more detailed diagram of camera 300 coupled to the Internet 150 is shown. Camera 300 is coupled to an external modem 101. Camera 300 is coupled to modem 101 via any of several communications means (e.g., Universal Serial Bus, infrared link, and the like). Modem 101 is in turn coupled to a POTS telephone jack 102 at the camera's location. Telephone jack 102 couples modem 101 to one of modems 103 of ISP 110 via the telephone company's local loop. ISP 110 is directly coupled to the Internet 150 via an all digital connection (e.g., a T1 line).

Continuing with reference to FIG. 1C, modem 101 is shown as an external modem. However, the functionality of modem 101 can be implemented directly within the electronics of camera 300 (e.g., via a modem application-specific integrated circuit or ASIC), or alternatively can be implemented as a software-only modem executing on a computer within camera 300. As such, it should be appreciated that, at the hardware connectivity level, modem 101 can take several forms. For example, a wireless modem can be used in which case the camera is not connected via an external wire to any land line. Alternatively, there may be applications in which camera 300 includes suitable electronic components enabling a connection to a conventional computer system network (e.g., Ethernet, AppleTalk, and the like), which is in turn directly connected to the Internet (e.g., via a gateway, a firewall, and the like), thereby doing away with the requirement for an ISP. Hence, it should be appreciated that the present invention is not limited to any particular method of accessing the Internet 150.

Refer now to FIG. 2 which illustrates server computer system 190 upon which embodiments of the present invention may be practiced (the following discussion is also pertinent to a client computer system). In general, server computer system 190 comprises bus 200 for communicating information, processor 201 coupled with bus 200 for processing information and instructions, random access memory 202 coupled with bus 200 for storing information and instructions for processor 201, read-only memory 203 coupled with bus 200 for storing static information and instructions for processor 201, data storage device 204 such as a magnetic or optical disk and disk drive coupled with bus 200 for storing information and instructions, optional display device 205 coupled to bus 200 for displaying information to the computer user, optional alphanumeric input device 206 including alphanumeric and function keys coupled to bus 200 for communicating information and command selections to processor 201, optional cursor control device 207 coupled to bus 200 for communicating user input information and command selections to processor 201, and network interface card (NIC) 208 coupled to bus 200 for communicating from a communication network to processor 201.

Display device 205 of FIG. 2 utilized with server computer system 190 of the present invention may be a liquid crystal device, cathode ray tube, or other display device

suitable for creating graphic images and alphanumeric characters recognizable to the user. Cursor control device 207 allows the computer user to dynamically signal the two-dimensional movement of a visible symbol (pointer) on a display screen of display device 205. Many implementations of the cursor control device are known in the art including a trackball, mouse, joystick or special keys on alphanumeric input device 206 capable of signaling movement of a given direction or manner of displacement. It is to be appreciated that the cursor control 207 also may be directed and/or activated via input from the keyboard using special keys and key sequence commands. Alternatively, the cursor may be directed and/or activated via input from a number of specially adapted cursor directing devices.

Referring now to FIG. 3, a block diagram of digital camera 300 is shown for use in accordance with the present invention. Camera 300 preferably comprises imaging device 314, system bus 316 and computer 318. Imaging device 314 is optically coupled to object 312 and electrically coupled via system bus 316 to computer 318. Once a photographer has focused imaging device 314 on object 312 and, using a capture button or some other means, instructed camera 300 to capture an image of object 312, computer 318 commands imaging device 314 via system bus 316 to capture raw data representing object 312. The captured raw data are transferred over system bus 316 to computer 318, which performs various processing functions on the data before storing it in memory. System bus 316 also passes various status and control signals between imaging device 314 and computer 318.

Referring now to FIG. 4, a block diagram of one embodiment of computer 318 is shown. System bus 316 provides connection paths between imaging device 314, an optional power manager 442, central processing unit (CPU) 444, dynamic random-access memory (DRAM) 446, input/output interface (I/O) 448, non-volatile memory 450, and buffers/connectors 452. Removable memory 454 connects to system bus 316 via buffers/connectors 452. Alternatively, camera 300 may be implemented without removable memory 454 or buffers/connectors 452.

Power manager 442 communicates via line 466 with power supply 456 and coordinates power management operations for camera 300. CPU 444 typically includes a conventional processor device for controlling the operation of camera 300. In the present embodiment, CPU 444 is capable of concurrently running multiple software routines to control the various processes of camera 300 within a multi-threaded environment. DRAM 446 is a contiguous block of dynamic memory which may be selectively allocated to various storage functions. LCD controller 490 accesses DRAM 446 and transfers processed image data to LCD screen 402 for display.

I/O 448 is an interface device allowing communications to and from computer 318. I/O 448 permits an external device (not shown) to connect to and communicate with computer 318. I/O 448 also interfaces with a plurality of buttons and/or dials 404, and optional status LCD 406, which in addition to LCD screen 402, are the hardware elements of the camera's user interface 408.

Non-volatile memory 450, which may typically comprise a conventional read-only memory or flash memory, stores a set of computer-readable program instructions to control the operation of camera 300.

Referring now to FIG. 5, a memory map showing one embodiment of dynamic random access memory (DRAM) 446 is shown. In the present embodiment, DRAM 446 includes RAM disk 532, system area 534, and working memory 530.

RAM disk 532 is a memory area used for storing raw and compressed data and typically is organized in a "sectored" format similar to that of conventional hard disk drives. In the present embodiment, RAM disk 532 uses a well-known and standardized file system to permit external devices, via I/O 448 of FIG. 4, to readily recognize and access the data stored on RAM disk 532. System area 534 typically stores data regarding system errors (for example, why a system shutdown occurred) for use by CPU 444 (FIG. 4) upon a restart of computer 318 (FIG. 3).

Working memory 530 includes various stacks, data structures and variables used by CPU 444 while executing the software routines used within computer 318. Working memory 530 also includes several input buffers 538 for temporarily storing sets of raw data received from imaging device 314 (FIG. 3), and frame buffer 536 for storing data for display on LCD screen 402 (FIG. 4). In the present embodiment, each input buffer 538 and frame buffer 536 are split into two separate buffers (shown by the dashed lines) to improve the display speed of the digital camera and to prevent the tearing of the image in LCD screen 402.

Referring now to FIG. 6, a diagram of the connectivity and application software of camera 300 of FIG. 3 is shown. At the software level, computer 318 (FIG. 3) of camera 300 hosts any network protocol that supports a persistent network connection. This coupling can be accomplished over any network protocol that supports a persistent network connection, such as TCP/IP (Transmission Control Protocol/Internet Protocol) including Point-to-Point Protocol, NetBIOS, IPX, and LU6.2, and link layers protocols such as Ethernet, token ring, and ATM. Protocol stack 601 interfaces with the communications hardware 602 (e.g., a modem) of camera 300 and the application layer 603. The bottom of protocol stack 601 includes communication hardware interface drivers which interface directly with the various communications hardware with which camera 300 must function (e.g., a Universal Serial Bus and the like). Applications layer 603, protocol stack 601, and operating system 604 are installed as software modules in DRAM 446 (FIG. 4) of camera 300. Software applications within applications layer 603 interface with operating system 604. Operating system 604 controls the hardware functionality of camera 300 (e.g., taking pictures, storing pictures, and the like) via camera hardware 605. Incoming data 620, such as HTTP (Hypertext Transfer Protocol) requests and the like, are received and outgoing data 630, such as HTML (Hypertext Markup Language) files and the like, are transferred to and from camera 300 via protocol stack 601 and communications hardware 602. Web browser 121 of FIG. 1A (or a program of similar function) can process data files, launch plug-ins, and run Java applets that communicate with camera 300 in a variety of methods in addition to those methods involving an exchange of HTML files.

FIG. 7 illustrates an executable program 700 for client computer system 120 (specifically, Web browser 121 of FIG. 1A or a program of similar function) to remotely access camera 300 (FIG. 3), where executable program 700 is implemented in accordance with the present invention as program instructions stored in computer-readable memory units (e.g., read-only memory 203) and implemented by processor 201 of server computer system 190 of FIG. 1A (specifically, by Web server 161). Executable program 700 performs functions both for and in response to Web browser 121 (or a program of similar function) and camera 300. The description below first discusses the steps associated with setting up a connection between executable program 700 and camera 300, then discusses the steps associated with

11

setting up a connection between executable program 700 and Web browser 121 (or a program of similar function), however, the present invention is not limited by the order in which these steps are presented.

In the present embodiment, executable program 700 is identified and accessed by its own unique address, commonly referred to as an URL (Unified Resource Locator), as is well known in the art. The URL for executable program 700 fully describes where it resides on a communication network (e.g., the Internet 150) and how it is accessed. In the present embodiment, included in the URL for executable program 700 is the name of camera 300. Accordingly, in one embodiment using a servlet for executable program 700, a standard format for a URL is: `http://webserverHostName/cameraServletWellKnownName/cameraName`.

In step 705, executable program 700 receives and accepts a connection request from camera 300. Executable program 700 runs constantly on Web server 161 and is configured to listen for connection requests on a plurality of communication protocols (e.g., TCP, NetBIOS, and the like). In the present embodiment, camera 300 is connected to executable program 700 via Web server 161 as described in conjunction with FIG. 1A, and opens the connection using the connectivity and application software described in conjunction with FIG. 6.

In step 710, executable program 700 receives and reads registration information from camera 300. As mentioned above, executable program 700 is configured to communicate using a number of different communication protocols. Such registration information includes the name of the camera and authentication information such as security information and account information. Executable program 700 uses the camera name to identify the camera and locate it in response to a client request.

In step 720, executable program 700 compares the registration information with a predefined access control list to determine if camera 300 is a camera for which Web server 161 is to provide support and service. If not, executable program 700 closes the connection between Web server 161 and camera 300.

In step 725, upon successful completion of step 720, executable program 700 registers camera 300 and stores the camera's name and associated requirements, such as security and account information. Executable program 700 also sends a message that camera 300 is registered. At this point, the connection between executable program 700 can be either terminated or maintained at the option of the camera's operator. If the connection is terminated, the registration information is maintained by Web server 161 and can be later accessed by executable program 700 when a subsequent connection is made with camera 300.

In step 730, executable program 700 accepts the connection request from camera 300 and thus has a persistent and long term connection with camera 300. As described above, the connection can be an ongoing connection maintained from the time when camera 300 was first registered. New connections can be made in the future whenever camera 300 reinitiates the registration protocol. Once camera 300 and executable program 700 have established a connection, they then wait until a client also makes a connection to access the camera. However, as will be described below in conjunction with step 770, it is not necessary for there to be an open connection between the Web server and the camera at the same time that there is an open connection between the Web browser (or a program of similar function) and Web server to accomplish remote access of the camera in accordance with the present invention.

12

In step 740, executable program 700 receives and accepts a request for a connection from a client. The client enters the URL of executable program 700, including the name of the camera (e.g., camera 300) to which access is desired, into a Web browser (e.g., Web browser 121 on client computer system 120 of FIG. 1A or a program of similar function). Using standard communication protocols such as TCP/IP, Web server 161 is queried with the URL for executable program 700. Web server 161 recognizes the URL and makes the connection to executable program 700. Thus, the present invention establishes a single location identified by a known URL where the client always goes to connect to the camera, no matter where the camera is or where the client is. (For those cases in which TCP/IP is not available—for example, when the device is not attached to the Internet or the like—camera 300 connects to executable program 700 directly using a well-known address that is associated with the protocol being used, such as a NetBIOS name.)

In step 745, executable program 700 then assumes control over the TCP/IP connection between Web browser 121 (or a program of similar function) and Web server 161. Executable program 700 establishes a persistent and long term connection between the browser and server. That is, the connection between Web browser 121 (or a program of similar function) and Web server 161 is kept open by executable program 700. As will be described below in conjunction with step 770, it is not necessary for there to be an open connection between the client computer system and the Web server at the same time that there is an open connection between the Web server and the camera to accomplish remote access of the camera in accordance with the present invention.

In step 750, executable program 700 next determines if camera 300 is registered as discussed in conjunction with step 735. If camera 300 is not registered, executable program 700 sends an appropriate message to the client to indicate such.

In step 760, if the camera is registered, executable program 700 validates the required access information provided by the client against the security and account information provided when camera 300 was registered. For example, executable program 700 validates whether the client is utilizing an authorized password or user name. If not, executable program 700 transmits an appropriate message to the client. The present invention can optionally provide additional services related to security or account information. For example, it could control the type of access a client is permitted based on the authentication information received from the client, or it could verify credit information and bill the client for services requiring payment.

In step 765, upon satisfactory completion of step 760, executable program 700 extends the connection from Web browser 121 (or a program of similar function) to camera 300 if there is an established connection to the camera as described in conjunction with step 730. Hence, a client using Web browser 121 or a program of similar function has direct, remote access to camera 300 via executable program 700 in Web server 161.

In step 770, executable program 700 forwards commands from a client to camera 300, and forwards images and other data from camera 300 to the client via Web server 161 and Web browser 121 (or a program of similar function). That is, executable program 700 enables a direct communication between the client computer system and the camera allowing the client to remotely access and manage the camera. If the client and the camera are both concurrently connected to

13

executable program 700, then the client immediately receives the data, and camera 300 immediately executes any commands from the client.

However, if Web browser 121 (or a program of similar function) and camera 300 are not each connected at the same time, remote access to the camera is still accomplished in accordance with the present invention. If camera 300 is not on-line, a client uses Web browser 121 (or a program of similar function) to access Web server 161 and executable program 700. The client transmits commands, and executable program 700 stores the commands on Web server 161. The client may then close the connection to the Web server. Subsequently, when camera 300 opens a connection to Web server 161 and executable program 700, executable program 700 retrieves the commands and forwards them to the camera. Camera 300 downloads the commands and executes them. The results from executing the commands are then sent to executable program 700, which stores them on Web server 161 until they are retrieved by the client.

Similarly, if camera 300 establishes a connection with Web server 161 and executable program 700 but the client is not on-line, the camera can, for example, download images and any other data that executable program 700 stores on Web server 161. Camera 300 may then close the connection to the Web server. The client then later makes a connection to Web server 161 and executable program 700, which retrieves the data and forwards it to the client. The client can also enter commands at this time, which are stored by executable program 700 as described above.

In summary, with references to FIG. 1A and FIG. 7, the client, via Web browser 121 or a program of similar function, Web server 161 and executable program 700, accesses camera 300 to request and retrieve data. Web browser 121 or a program of similar function inserts the Internet address inside the data request (e.g., a HTTP request) and sends the request to Web server 161. Web server 161 receives the data request and associates the request with executable program 700, which in turn assumes the connection between Web server 161 and Web browser 121 (or a program of similar function), and which also establishes a connection between Web browser 121 (or a program of similar function) and camera 300. Executable program 700 subsequently forwards commands from the client to camera 300, and retrieves the requested data (e.g., a HTML file) containing the data and sends it back to Web browser 121 (or a program of similar function). Web browser 121 (or a program of similar function) then interprets the commands and displays the resulting image. The process of accessing a data file from a Web server is commonly referred to as accessing a Web page. Similarly, the process of sending data files from a Web server to a Web browser is commonly referred to as sending a Web page.

Thus, as described above, the present invention provides an intuitive and easy-to-use interface enabling remote access between a client and a camera. By functioning with widely used and familiar Web browsers (or programs of similar function) using standard format URLs to identify the executable program and camera, the present invention provides a simple and familiar interface for accessing the camera. By registering the camera and using an unchanging URL, name to identify the executable program and the camera, the present invention enables the client to locate and access the camera from any remote location no matter where the camera is located. In addition, by using a Java servlet or a cgi-bin for the executable program, the present invention is supported by commonly used Web servers and is readily implemented.

14

Executable program 700 is located on the Web server and not on camera 300, so it does not require additional and substantial memory dedicated to enabling remote access. As such, the present invention permits remote access within the constraints of the size of the camera. In addition, in accordance with the present invention, camera 300 does not require a separate, external computer system (e.g., a personal computer system) for connecting to ISP 110 (FIG. 1A) or for implementing commands and transmitting data, thus providing an inexpensive method for providing remote access to cameras.

Also, by locating the present invention on a Web server (e.g., Web server 161 of FIG. 1A), the Web server becomes a focal point for accessing and managing a plurality of cameras that otherwise would have to be managed and configured separately. For example, executable program 700 on Web server 161 could be updated with new software, and in effect all cameras accessed through that executable program would automatically be updated as well. As another example, executable program 700 could be configured to compile data regarding interactions between clients and all cameras accessed by the executable program. In another example, in accordance with the present invention, a client needs to go only to a single location to determine which of a plurality of cameras served by the executable program have data that have been downloaded to the Web server. Thus, instead of having to access a number of cameras separately, the present invention establishes a single location from which a client can access information about several cameras.

By functioning with a Web-based interface and widely used Web browsers (or programs of similar function), the present invention provides a simple, intuitive, and familiar interface for accessing camera 300's functionality. Accordingly, camera 300's controls and functions are intuitively easy to utilize. Since Web pages and their associated controls (e.g., push buttons, data entry fields, and the like) are very familiar to most users, the remote access functionality of camera 300 can be utilized without requiring a extensive learning period for new users. For example, a consumer purchasing a remotely accessible camera is typically able to easily and immediately use the remote accessibility functions with minimal set-up.

FIG. 8 illustrates a process 800 for remotely accessing camera 300 (FIG. 3), where process 800 is implemented as program instructions stored in computer-readable memory units (e.g., non-volatile memory 450 of FIG. 4) and implemented by CPU 444 (FIG. 4) of camera 300 in accordance with the present invention. FIG. 8 illustrates the process for remotely accessing a camera from the perspective of the actions performed by the camera and the camera operator in accordance with the present invention.

In step 805, camera 300 of FIG. 3 opens a connection to communication network 100 of FIG. 1A. In the present embodiment, camera 300 is coupled to server computer system 190 (specifically, Web server 161) as described in conjunction with FIG. 1A, and opens the connection using the connectivity and application software described in conjunction with FIG. 6. As described in conjunction with FIG. 1C, in the present embodiment, camera 300 couples directly to the telephone system such that a separate and dedicated computer system (e.g., a personal computer system) is not necessary.

In step 810, with the connection made to Web server 161, camera 300 requests a connection to executable program 700 (FIG. 7). For cases in which camera 300 is accessing

15

executable program 700 via, for example, a TCP/IP network such as the Internet 150, then executable program 700 is identified with a URL that is used by the camera to access the executable program. For those cases in which TCP/IP is not available (e.g., when the device is not attached to the Internet or the like), camera 300 connects to executable program 700 directly using a well-known address that is associated with the protocol being used, such as a NetBIOS name.

In step 820, with the camera connected to the executable program, camera 300 registers with executable program 700. For example, camera 300 provides information including an identification name and authentication information such as a password and account information. The information is electronically transmitted from camera 300 and read by executable program 700. Based on this information, the connection between camera 300 and executable program 700 is established if the camera is of the type that is designated to be supported by Web server 161.

In step 830, camera 300 and executable program 700 are linked via a persistent and long term connection; that is, the connection remains open awaiting a client to request access to the camera via Web server 161. As discussed above in conjunction with FIG. 7, it is not necessary for the camera and a client to be connected at the same time to executable program 700.

In step 835, camera 300 receives commands from and transmits data to a client using a Web browser or a program of similar function on a client computer system (e.g., client computer system 120 and Web browser 121 of FIG. 1A or a program of similar function). As described above in conjunction with FIG. 7, the commands and data can be transmitted through an active connection or stored on the Web server.

In the present embodiment, camera 300 is provided with several different operating modes for supporting various camera functions. In capture mode, camera 300 supports the actions of preparing to capture an image and of capturing an image. In review mode, camera 300 supports the actions of reviewing camera contents, editing and sorting images, and printing and transferring images. In play mode, camera 300 allows the client to view screen-sized images in the orientation that the image was captured. Play mode also allows the client to hear recorded sound associated to a displayed image, and to play back sequential groupings of images, which may comprise time lapse, slide show, and burst image images. The client preferably switches between the capture, review, and play modes.

Camera 300 is capable of implementing a wide variety of remote access and remote imaging/surveillance applications. In the present embodiment, camera 300 only records successive images for remote access by the client. The images are loaded into the camera's memory on a first-in, first-out (FIFO) basis, with the earliest recorded image being replaced by the latest recorded image. The number of images available to the client depends upon the amount of installed memory in the camera.

With reference still to step 835 of FIG. 8, when the client and the camera are connected to the Web server at the same time, the commands can be transmitted directly from Web browser 121 (or a program of similar function) to camera 300 via executable program 700 on Web server 161, and camera 300 then executes the commands on-line. Alternatively, when camera 300 is not connected to executable program 700, a client can store commands on Web server 161 by accessing the Web server in a normal fashion,

16

then entering the commands to be stored via executable program 700. Camera 300 then executes the commands when it subsequently connects with executable program 700.

Also in step 835, camera 300 transmits data to a client. Similar to the above, data from camera 300 can be transmitted directly to Web browser 121 or a program of similar function via executable program 700 on Web server 161, when the client and the camera are connected to the Web server at the same time. Alternatively, when a client is not connected to executable program 700, the data are stored on Web server 161 by executable program 700, which then retrieves and transmits the data to a client when the client subsequently connects with executable program 700.

This process of accessing camera 300 from Web browser 121 or a program of similar function occurs transparently with respect to the client. In a typical case, for example, the client types the URL for executable program 700 (which includes the name of camera 300) into Web browser 121 (or a program of similar function) and his "enter" or "return". In accordance with the present invention, the next Web page the client views is the image generated by the data returned from camera 300. Beyond entering the URL for executable program 700 including camera 300, no further action from the client is required in order to access the Web pages hosted by camera 300.

FIG. 9 illustrates a process 900 for remotely accessing camera 300 (FIG. 3), where process 900 is implemented as program instructions stored in computer-readable memory units (e.g., read-only memory) and implemented by the central processor of client computer system 120 (specifically, Web browser 121 or a program of similar function) of FIG. 1A. FIG. 9 illustrates the process for remotely accessing a camera from the perspective of the actions performed by the client in accordance with the present invention.

In step 905, the client opens a network connection such as a dial-up connection to ISP 115 of FIG. 1A.

In step 910, the client enters into a Web browser (e.g., Web browser 121 on client computer system 120 of FIG. 1A or a program of similar function) the URL of executable program 700, including the name of the camera (e.g., camera 300) to which access is desired. Alternatively, the client enters the URL of executable program 700 only. (For those cases in which TCP/IP is not available—for example, when the device is not attached to the Internet or the like—camera 300 connects to executable program 700 directly using a well-known address that is associated with the protocol being used, such as a NetBIOS name.) Using standard communication protocols such as TCP/IP, Web server 161 is queried with the URL for executable program 700. Web server 161 recognizes the URL and makes the connection to executable program 700. A persistent and long term two-way connection is opened between the client on Web browser 121 (or a program of similar function) and executable program 700.

In step 915, the client enters the authentication information required to gain access to executable program 700 and camera 300.

In step 920, as explained above, from his/her remote location the client sends commands to camera 300 and receives images from the camera. As explained above, the client and the camera do not have to be connected to the Web server at the same time to enable remote access. Depending on the particular application, the Web page for camera 300 can include control buttons, data entry fields, drop-down menus, and the like.

17

Thus, executable program 700 enables the client to access from a remote location the functional controls of camera 300 as well as the images and other data acquired by the camera.

Pseudo-Code Sections A, B, C, D and E below provide additional details regarding processes 700, 800 and 900 of FIGS. 7, 8 and 9. Pseudo-Codes Sections A through E represent the method of one embodiment of the present invention. However, it is appreciated that other embodiments are possible in accordance with the present invention.

18

With references to FIGS. 7 and 9, the pseudo-code for the connection of a client to the server computer system, specifically to executable program 700, is described in accordance with one embodiment of the present invention in Pseudo-Code Section A below.

Pseudo-Code Section A: Client Connection Setup

```

1. Client sends HTTP Post command to http://serverName/gateway device="deviceName"
   Note: authentication/security data is optional and is handled by the browser.
2. Client receives HTTP response from Gateway
3. If response is OK or OKDataPresent
{
    Client has TCP connection it can use to send/receive anything to/from the device
    The protocol used can be any that the Client & Device agree upon.
}
else
{
    The response is failed or DataPresent.
    The TCP connection is closed:
}

```

Note: the OKDataPresent and DataPresent responses are informational. The Client at any time on separate connections may issue a CacheData or GetCache command to the Gateway. See algorithms C.

30 With references to FIGS. 7 and 8, the pseudo-code for the connection of a device (e.g., digital camera 300 of FIG. 1A) to the server computer system, specifically to executable program 700, is described in accordance with one embodiment of the present invention in Pseudo-Code Section B below.

Pseudo-Code Section B: Device Connection Setup/Registration

```

1. Device establishes a connection to the Gateway using any networking protocol
   supported by the Gateway (TCP, NetBIOS, IPX, 802.2, etc) using a known address.
2. Device sends a Register request to the Gateway on the connection passing
   its name (and optionally authentication/security information).
3. Device receives response from Gateway
4. If the response is OK or OKDataPresent
{
    The Device may use the existing connection to wait for requests from clients
    or the Device may optionally close the connection.
}
else
{
    The response is Failed or DataPresent
    The connection is closed.
}

```

Note: the OKDataPresent and DataPresent responses are informational. The Device at any time on separate connections may issue a CacheData or GetCache command to the Gateway. See algorithm D.

60

With reference to FIG. 7, the pseudo-code for executable program 700 in response to a client connection request is described in accordance with one embodiment of the present invention in Pseudo-Code Section C below.

65 Pseudo-Code Section C: Gateway Handles Client Connection Request

```

Start:
    Wait for incoming requests from clients
    if its an HTTP Post command with parameters CacheData and deviceName
    {
        go to CacheData
    }
    else if its an HTTP Get command with parameters GetCache and deviceName
    {
        go to GetCache
    }
    if its an HTTP Post command from a client with the name of device
    {
        go to Connect
    }
    else
    {
        go to Fail
    }
Connect:
    Look up device by name in registry
    if the device entry is found
    {
        if there is a device connection id in the entry
        {
            if the entry is not busy
                go to Success
            else go to Fail
        }
        else
        {
            Attempt to establish connection with device at last known address
            if connection established
            {
                go to Success
            }
            else go to Fail
        }
    }
    else
    {
        send Fail response
        close connection
        go to Start
    }
Fail:
    if there is data in the cache
    {
        set response to DataPresent
    }
    else
    {
        set response to Fail
    }
    send response
    close connection
    go to Start
Success:
    if there is data in the cache
    {
        set response to OkDataPresent
    }
    else
    {
        set response to Ok
    }
    store connection id of the client in the registry entry
    go to Start
CacheData:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        store data in Cache for device with identification information

```

-continued

```

    send Ok response
    close connection
    go to Start;
}
GetCache:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        send data in Cache for this client
        close connection
        go to Start
    }
}

```

With reference to FIG. 7, the pseudo-code for executable program 700 in response to a device connection request is described in accordance with one embodiment of the present invention in Pseudo-Code Section D below.

Pseudo-Code Section D: Gateway Handles Device Requests

```

Start:
    Wait for incoming requests from devices
    Receive incoming request
    if its a Register request
    {
        go to Register
    }
    else if its a CacheData request
    {
        go to CacheData
    }
    else if its a GetCache request:
    {
        go to GetCache
    }
    else
    {
        return Failed
        close connection
        go to Start
    }
Register:
    Note: The gateway may optionally authenticate device
    Get device name from Register request
    look up name in table
    if name is found
    {
        go to Success
    }
    else
    {
        The gateway may optionally add the name or reject the
        registration attempt, then go to Success or Fail
        respectively
    }
Success:
    check for the presence of cached client data
    if cached data is present
    {
        set response to OkDataPresent;
    }
    else
    {
        set response to Ok
    }
}

```

-continued

```

    Store a connection id for the incoming connection in
    the registry entry for the device.
    send response
    go to Start
Fail:
    if cached data from clients is present for this device
    {
        set response to DataPresent
    }
    else
    {
        set response to Fail
    }
    send response
    close connection
    go to Start
CacheData:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        store data in Cache for client
        send Ok response
        close connection
        go to Start;
    }
GetCache:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        send data in Cache for device to the device
        close connection
        go to Start
    }
}

```

With reference to FIG. 7, the pseudo-code for executable program 700, specifically the handling of data when an open connection is present between the device and the client

23

computer system, is described in accordance with one embodiment of the present invention in Pseudo-Code Section E below.

Pseudo-Code Section E: Gateway Handles Data on Existing Connections

```

Start:
  Wait for data
  Map the incoming connection id to its partner connection id
  if there is incoming data
  {
    Send the data out on the partner connection id
    go to Start
  }
  else if the connection closed
  {
    if it is a client connection
    {
      Remove the client's connection id from the
      device's entry in the registry.
    }
    else
    {
      Remove both the client and device connection ids
      from the device's entry in the registry.
      Close the client's connection
      go to Start
    }
  }
  go to Start
  
```

As described above, the remote accessibility of camera **300** provides for many new applications of digital imagery. One such application involves setting up camera **300** at some remote location and using it to take pictures at successive intervals. These pictures would be accessed via the Internet **150** as they are taken. The interval can be adjusted (e.g., more or less pictures per minute) in response to commands entered by a client via a Web browser (e.g., Web browser **121** of FIG. 1A or a program of similar function).

Another application involves using camera **300** in conjunction with a motion detector. When used in conjunction with a motion detector, camera **300** can be configured to capture an image in response to receiving a signal from the motion detector (e.g., detecting the motion of an intruder), thereby taking a picture of whatever triggered the detector's signal output. Alternatively, camera **300** can detect motion by simply comparing successive images to detect changes between them, thereby dispensing with the need for a separate motion detector.

Yet another application involves using camera **300** in conjunction with a remote aiming device. Camera **300** can be mounted on a remotely operated aiming device (e.g., a motorized tripod). The aiming device is controlled via the Internet **150** in the same manner the camera is controlled via the Internet **150**. Alternatively, camera **300** could be coupled to control the remote aiming device directly. The remote aiming device allows a client to control the field of view of the camera **300** in the same manner the client controls other functionality (e.g., picture resolution, picture interval, and the like).

In this manner, executable program **700** of the present invention is able to implement sophisticated remote surveillance of the type previously performed by expensive, prior art closed circuit television devices. Unlike the prior art, however, executable program **700** is inexpensive and relatively simple to implement.

Thus, the present invention provides a method for making a digital camera and its internally stored data remotely

24

accessible. The present invention enables the digital camera to be set to continuously take pictures of scenes and items of interest and to allow a user to access those pictures at any time. The present invention implements remote accessibility via a communication network such as the Internet, thus allowing the user to access the digital camera from virtually an unlimited number of locations and with the camera in virtually any location.

A digital camera in accordance with the present invention does not require a separate, external computer system (e.g., a personal computer system) for Internet connectivity, thus providing an inexpensive method for making remotely accessible digital cameras widely available. In addition, a digital camera in accordance with the present invention is accessed via the widely used and very familiar Web browser (or a program of similar function). By functioning with typical, widely used Web browsers (or programs of similar function), the present invention provides a simple, intuitive, and familiar interface for remotely accessing the digital camera's functionality and capabilities. In so doing, the controls and functions of the digital camera are intuitively easy to utilize, and do not require an extensive learning period for new users. The present invention also provides an efficient and user-transparent process of obtaining the address of a digital camera. Also, the present invention provides a method for efficiently administering a plurality of separate digital cameras.

Although the present invention is described in the context of a digital camera, it is not limited to this embodiment. Hence, the present invention does not provide only a limited degree of functionality as in the prior art applications; that is, it is not limited to either chat or video conferencing, or the like. As such, the present invention is capable of establishing a connection between any type of client system and remote device.

The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.

What is claimed is:

1. A method for allowing a client computer to remotely access a digital image capture unit via a communication network wherein a server computer system and said client computer system are communicatively coupled with communication equipment, said method comprising the steps of:
 - a) allowing said digital image capture unit to establish communication with said server computer system over said network, wherein said digital image capture unit includes connectivity software that enables said digital image capture unit to establish a network connection with said server computer system;
 - b) receiving an address of said digital image capture unit and registering said address in an executable program on said server computer system, wherein said digital image capture unit automatically registers said address with said server computer system;
 - c) allowing said client computer system to access said executable program on said server computer system; and

25

- d) establishing direct communication between said client computer system and said digital image capture unit by communicating commands between said client computer system and said digital image capture unit via said executable program, such that data captured by said digital image capture unit is transferred to said client computer system via said server computer system.
2. The method of claim 1 wherein said communication network is the Internet.
3. The method of claim 1 wherein said communication network is a Local Area Network.
4. The method of claim 1 wherein said image capture unit is a digital camera.
5. The method of claim 1 wherein step a) further comprises communicating authentication information between said digital image capture unit and said executable program.
6. The method of claim 1 wherein said executable program is a Java servlet.
7. The method of claim 1 wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).
8. The method of claim 1 wherein step c) further comprises connecting said digital image capture unit to said server computer system via a Local Area Network.
9. The method of claim 1 wherein step c) further comprises connecting said digital image capture unit to said server computer system via an input/output port of said server computer system.
10. The method of claim 1 wherein step c) further comprises connecting said digital image capture unit to said server computer system via the Internet.
11. The method of claim 1 wherein step d) further comprises storing said commands in a memory unit of said server computer system and communicating said commands to said digital image capture unit at a time when a connection is made between said server computer system and said digital image capture unit.
12. The method of claim 1 further comprising the steps of:
- e) accessing via said executable program data acquired by said digital image capture unit; and
 - f) transferring said data from said digital image capture unit to said client computer system via said server computer system.
13. The method of claim 12 further comprising storing said data in a memory unit of said server computer system and communicating said data to said client computer system at a time when a connection is made between said server computer system and said client computer system.
14. A computer system comprising:
- a) a processor coupled to a bus; and
 - a memory unit coupled to said bus and having stored therein an executable program that when executed by said processor implements a method for allowing a client computer to remotely access a digital image capture unit via a communication network, said method comprising the steps of:
 - a) allowing said digital image capture unit to establish communication with said server computer system over said network, wherein said digital image capture unit includes connectivity software that enables said digital image capture unit to establish a network connection with said server computer system;
 - b) receiving an address of said digital image capture unit and registering said address with said executable program, wherein said digital image capture unit automatically registers said address with said server computer system;

26

- c) allowing said client computer system to access said executable program over said communication network; and
 - d) establishing direct communication between said client computer system and said digital image capture unit by communicating commands between said client computer system and said digital capture unit via said executable program.
15. The computer system of claim 14 wherein said computer system is a server computer system.
16. The computer system of claim 14 wherein said digital image capture unit is a digital camera.
17. The computer system of claim 14 wherein said executable program is a Java servlet.
18. The computer system of claim 14 wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).
19. The computer system of claim 14 wherein said digital image capture unit and said computer system are connected via a Local Area Network.
20. The computer system of claim 14 wherein said digital image capture unit and said computer system are connected via an input/output port of said computer system.
21. The computer system of claim 14 wherein said digital image capture unit and said computer system are connected via the Internet.
22. In a communication network comprising a server computer system and a client computer system communicatively coupled with communication equipment, a method for allowing said client computer to remotely access a digital camera via said communication network, said method comprising the steps of:
- a) allowing said digital camera to establish communication with said server computer system over said communication network, wherein said digital camera includes connectivity software that enables said digital camera to establish a network connection with said server computer system;
 - b) receiving an address of said digital camera and registering said address in an executable program on said server computer system, wherein said digital camera automatically registers said address with said server computer system;
 - c) allowing said client computer system to access said executable program on said server computer system;
 - d) establishing direct communication between said client computer system and said digital camera by communicating commands between said client computer system and said digital camera via said executable program;
 - e) accessing via said server computer system data acquired by said digital camera; and
 - f) transferring said data from said digital camera to said client computer system via said server computer system.
23. The method of claim 22 wherein said executable program is a Java servlet.
24. The method of claim 22 wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).
25. A method for allowing a client computer to remotely access a digital image capture device via a communication network wherein a server computer system and said client computer system are communicatively coupled with communication equipment, said method comprising the steps of:
- a) allowing said digital image capture device to establish communication with said server computer system over

27

said communication network, wherein said digital image capture device includes connectivity software that enables said digital image capture device to establish a network connection with said server computer system;

- b) receiving an address of said digital image capture device and registering said address in an executable program on said server computer system, wherein said digital image capture device automatically registers said address with said server computer system;
- c) allowing said client computer system to access said executable program on said server computer system; and

28

- d) establishing direct communication between said client computer system and said digital image capture device by communicating commands between said client computer system and said digital image capture device via said executable program, such that data from said digital image capture device is transferred to said client computer system via said server computer system.

26. The method of claim **25** wherein said executable program is a Java servlet.

27. The method of claim **25** wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).

* * * * *



US006662218B2

(12) **United States Patent**
Mighdoll et al.

(10) Patent No.: **US 6,662,218 B2**
(45) Date of Patent: ***Dec. 9, 2003**

(54) **METHOD OF TRANSCODING DOCUMENTS
IN A NETWORK ENVIRONMENT USING A
PROXY SERVER**

(75) Inventors: **Lee S. Mighdoll**, San Francisco, CA
(US); **Bruce A. Leak**, Palo Alto, CA
(US); **Stephen C. Perlman**, Mountain
View, CA (US); **Phillip Y. Goldman**,
Los Altos, CA (US)

(73) Assignee: **WebTV Networks, Inc.**, Mountain
View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 0 days.

This patent is subject to a terminal dis-
claimer.

(21) Appl. No.: **10/247,885**

(22) Filed: **Sep. 20, 2002**

(65) **Priority Publication Data**

US 2003/0014499 A1 Jan. 16, 2003

Related U.S. Application Data

(63) Continuation of application No. 09/343,067, filed on Jun.
29, 1999, now abandoned, which is a continuation of application
No. 08/656,924, filed on Jun. 3, 1996, now Pat. No.
5,918,013.

(51) **Int. Cl.** **G06F 15/16**

(52) **U.S. Cl.** **709/219; 709/203; 709/217;
709/228; 709/229; 709/246**

(58) **Field of Search** **709/200-203,
709/217-219, 227-229, 231-232, 246-247**

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,575,579 A 3/1986 Simon et al. 178/4
4,852,151 A 7/1989 Dittakavi et al. 379/97

4,922,523 A 5/1990 Hashimoto 379/96
4,975,944 A 12/1990 Cho 379/209
4,995,074 A 2/1991 Goldnan et al. 379/97
5,005,011 A 4/1991 Perlman et al. 340/728
5,095,494 A 3/1992 Takahashi et al. 375/10
5,220,420 A 6/1993 Hoarty et al. 358/86
5,241,587 A 8/1993 Horton et al. 379/92
5,263,084 A 11/1993 Chaput et al. 379/215
5,287,401 A 2/1994 Lin 379/98
5,299,307 A 3/1994 Young 395/161

(List continued on next page.)

OTHER PUBLICATIONS

Rosoff, Matt, Review: "Gateway Destination PC," c/net
inc., 2 pages, Feb. 19, 1996.

Seidman, Robert, Article: "What Larry and Lou Know (That
You Don't)," c/net inc., 2 pages, Jan. 29, 1996.

Stellin, Susan, Article: "The \$500 Web Box: Less is More?"
c/net inc., 2 pages, 1996.

(List continued on next page.)

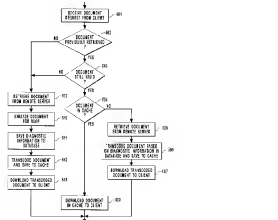
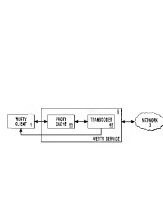
Primary Examiner—Bharat Barot

(74) Attorney, Agent, or Firm—Workman, Nydegger

(57) **ABSTRACT**

A method is described of providing a document to a client
coupled to a server. The server functions as a proxy on
behalf of the client for purposes of accessing a remote
server. In the method, a document is retrieved from the
remote server in response to a request from the client. The
document includes data to be used by the client in generating
a display. The proxying server alters (i.e., transcodes) the
data in the document to form a transcoded document. The
transcoded document is then transmitted to the client. The
proxying server transcodes the data in the document in order
to perform at least one of the following functions: (1)
matching decompression requirements at the client; (2)
converting the document into a format compatible for the
client; (3) reducing latency experienced by the client; and
(4) altering the document to fit into smaller memory space.

31 Claims, 12 Drawing Sheets



U.S. PATENT DOCUMENTS

5,325,423 A	6/1994	Lewis	379/60	5,678,041 A	10/1997	Baker et al.	395/609
5,329,619 A	7/1994	Page et al.	395/200	5,727,159 A	3/1998	Kikinis	395/200.76
5,341,293 A	8/1994	Verletney et al.	364/419.17	5,842,216 A	11/1998	Anderson et al.	707/203
5,369,688 A	11/1994	Tsukamoto et al.	379/100	5,889,955 A	3/1999	Shinozaki et al.	395/200.54
5,469,540 A	11/1995	Powers, III et al.	395/158	5,918,013 A	* 6/1999	Mighdoli et al.	709/217
5,488,411 A	1/1996	Lewis	348/8	5,978,817 A	11/1999	Gianandrea et al.	707/501
5,490,208 A	2/1996	Remillard	379/96	5,996,022 A	* 11/1999	Krueger et al.	709/247
5,530,852 A	6/1996	Meske, Jr. et al.	395/600	6,018,619 A	1/2000	Allard et al.	395/200.54
5,538,255 A	7/1996	Barker	463/41	6,226,412 B1	5/2001	Schwab	382/232
5,558,339 A	9/1996	Periman	463/42	OTHER PUBLICATIONS			
5,561,709 A	10/1996	Remillard	379/96	<i>Administrator's Guide, Netscape Proxy Server Version 2.0,</i>			
5,564,001 A	10/1996	Lewis	395/154	Netscape Communications Corporation, pp. 19–20, 1996.			
5,572,643 A	11/1996	Judson	395/793	Chankhuthod, Anawat, et al., "A Hierarchical Internet			
5,586,257 A	12/1996	Periman	463/42	<i>Object Cache,</i> " 1996 USEWIX Technical Conference (6			
5,586,260 A	12/1996	Hu	395/200.2	pages).			
5,612,730 A	3/1997	Lewis	348/8	Farrow, Rik, "Securing the Web: fire walls, proxy servers,			
5,623,600 A	4/1997	Ji et al.	395/187.01	<i>and data driven attacks,</i> " InfoWorld, Jun. 19, 1995, vol. 7,			
5,654,886 A	8/1997	Zereski, Jr. et al.	364/420	No. 25, pp. 103–104.			
5,657,390 A	8/1997	Elgarni et al.	380/49	* cited by examiner			
5,657,450 A	8/1997	Rao et al.	395/610				

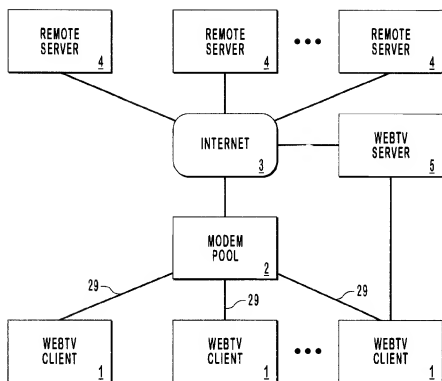


FIG. 1

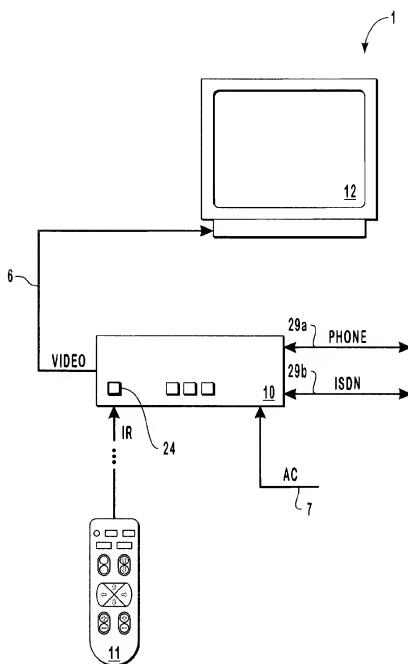


FIG. 2

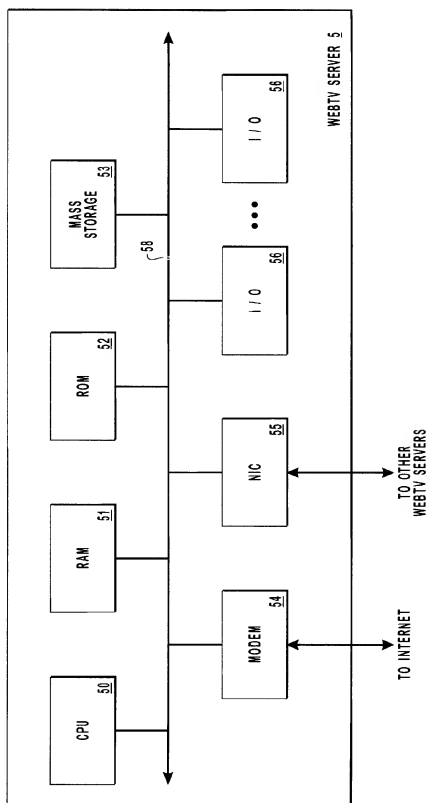


FIG. 3

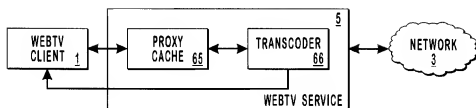


FIG. 4A

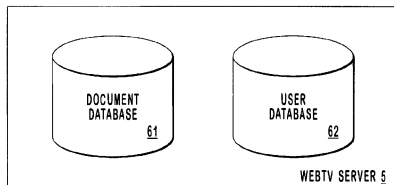


FIG. 4B

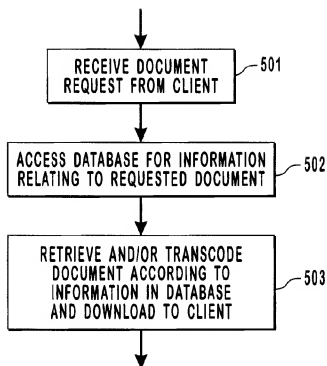


FIG. 5

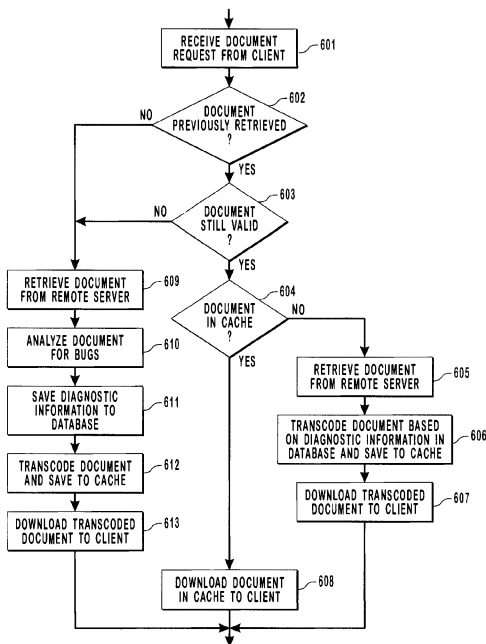


FIG. 6

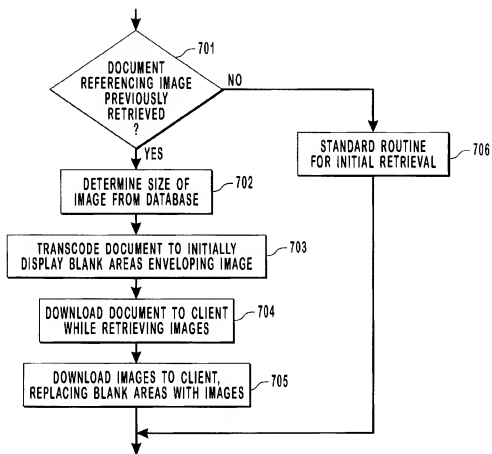


FIG. 7

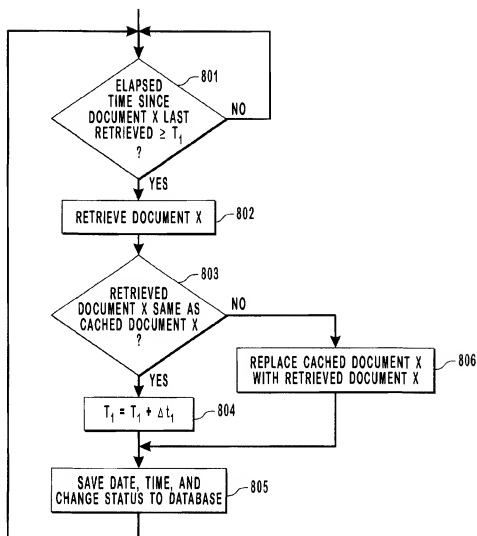


FIG. 8

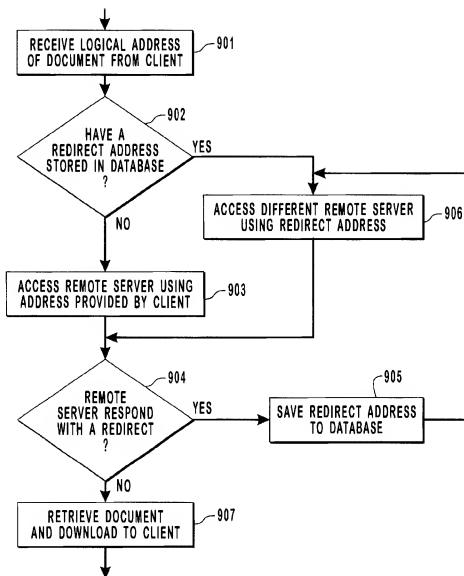


FIG. 9

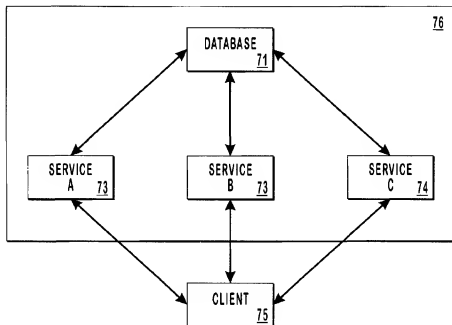


FIG. 10
(PRIOR ART)

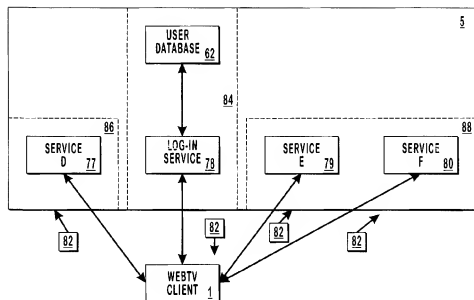


FIG. 11

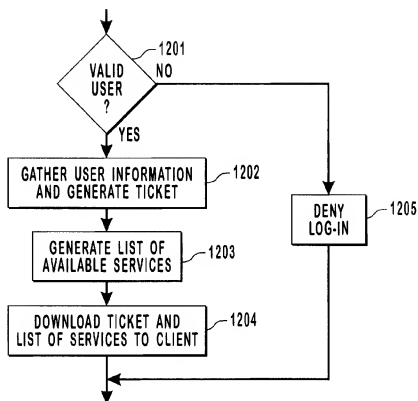


FIG. 12

1

METHOD OF TRANSCODING DOCUMENTS IN A NETWORK ENVIRONMENT USING A PROXY SERVER

RELATED APPLICATION

This is a continuation of U.S. patent application Ser. No. 09/343,067, entitled "Method of Transcoding Documents in a Network Environment Using a Proxy Server," filed Jun. 29, 1999, now abandoned which is a continuation of U.S. application Ser. No. 08/656,924 entitled "Method of Transcoding Documents in a Network Environment Using a Proxy Server," filed Jun. 3, 1996, now U.S. Pat. No. 5,918, 013 both of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. The Field of the Invention

The present invention pertains to the field of client-server computer networking. More particularly, the present invention relates to a method of transcoding documents in a network environment using a proxy server.

2. The Prior State of the Art

The number of people using personal computers has increased substantially in recent years, and along with this increase has come an explosion in the use of the Internet. One particular aspect of the Internet which has gained widespread use is the World-Wide Web ("the Web"). The Web is a collection of formatted hypertext pages located on numerous computers around the world that are logically connected by the Internet. Advances in network technology and software providing user interfaces to the Web ("Web browsers") have made the Web accessible to a large segment of the population. However, despite the growth in the development and use of the Web, many people are still unable to take advantage of this important resource.

Access to the Web has been limited thus far mostly to people who have access to a personal computer. However, many people cannot afford the cost of even a relatively inexpensive personal computer, while others are either unable or unwilling to learn the basic computer skills that are required to access the Web. Furthermore, Web browsers in the prior art generally do not provide the degree of user-friendliness desired by some people, and many computer novices do not have the patience to learn how to use the software. Therefore, it would be desirable to provide an inexpensive means by which a person can access the Web without the use of a personal computer. In particular, it would be desirable for a person to be able to access the Web pages using an ordinary television set and a remote control, so that the person feels more as if he or she is simply changing television channels, rather than utilizing a complex computer network.

Prior art Web technology also has other significant limitations which can make a person's experience unpleasant when browsing the Web. Web documents are commonly written in HTML (Hypertext Mark-up Language). HTML documents sometimes contain bugs (errors) or have features that are not recognized by certain Web browsers. These bugs or quirks in a document can cause a Web browser to fail. Thus, what is needed is a means for reducing the frequency with which client systems fail due to bugs or quirks in HTML documents.

Another problem associated with browsing the Web is latency. People commonly experience long, frustrating delays when browsing the Web. It is not unusual for a person to have to wait minutes after selecting a hypertext link for a

2

Web page to be completely downloaded to his computer and displayed on his computer screen. There are many possible causes for latency, such as heavy communications traffic on the Internet and slow response of remote servers. Latency can also be caused by Web pages including images. One reason for this effect is that, when an HTML document references an image, it takes time to retrieve the image itself after the referencing document has been retrieved. Another reason is that, in the prior art, if the referencing document does not specify the size of the image, the client system generally cannot display the Web page until the image itself has been retrieved. Numerous other sources of latency exist with respect to the Web. Therefore, what is needed is a means for reducing such latency, to eliminate some of the frustration which typically has been associated with browsing the Web.

Security is another concern associated with the Internet. Internet service providers (ISPs) generally maintain certain information about each customer in a database. This information may include information which a customer may not wish to become publicly known, such as social security numbers and credit card numbers. Maintaining the confidentiality of this information in a system that is connected to an expensive publicly-accessible computer network like the Internet can be problematic. Further, the problem can be aggravated by the fact that an ISP often provides numerous different services, each of which has access to this database. Allowing access to the database by many different entities creates many opportunities for security breaches to occur. Therefore, what is needed is a way to improve the security of confidential customer information in a server system coupled to the Internet.

SUMMARY AND OBJECTS OF THE INVENTION

A method is described of providing a document to a client coupled to a server. The server functions as a proxy on behalf of the client for purposes of accessing a remote server. In the method, a document is retrieved from the remote server in response to a request from the client. The document includes data to be used by the client in generating a display. The proxying server alters (i.e., transcodes) the data in the document to form a transcoded document. The transcoded document is then transmitted to the client.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follows. The proxying server transcodes the data in the document in order to perform at least one of the following functions: (1) matching decompression requirements at the client; (2) converting the document into a format compatible for the client; (3) reducing latency experienced by the client; and (4) altering the document to fit into smaller memory space.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follow.

BRIEF DESCRIPTION OF THE DRAWINGS

In order that the manner in which the above-recited and other advantages and objects of the invention are obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered limiting of its scope, the invention will be

3

described and explained with additional specificity and detail through the use of the accompanying drawings in which:

FIG. 1 illustrates several clients connected to a proxying server in a network;

FIG. 2 illustrates a client according to the present invention;

FIG. 3 is a block diagram of a server according to the present invention;

FIG. 4A illustrates a server including a proxy cache and a transcoder;

FIG. 4B illustrates databases used in a server according to the present invention;

FIG. 5 is a flow diagram illustrating a routine for transcoding a document retrieved from a remote server using data stored in a persistent database;

FIG. 6 is a flow diagram illustrating a routine for transcoding an HTML document for purposes of eliminating bugs or undesirable features;

FIG. 7 is a flow diagram illustrating a routine for reducing latency when downloading a document referencing an image to a client;

FIG. 8 is a flow diagram illustrating a routine for updating documents stored in the proxy cache using data stored in a persistent database;

FIG. 9 is a flow diagram illustrating a routine used by a server for retrieving documents from another remote server;

FIG. 10 is a block diagram of a prior art server system showing a relationship between various services and a database;

FIG. 11 is a block diagram of a server system according to the present invention showing a relationship between various services and a user database; and

FIG. 12 is a flow diagram illustrating a routine used by a server for regulating access to various services provided by the server.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

A method and apparatus are described for providing proxying and transcoding of documents in a network. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be evident, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

The present invention includes various steps, which will be described below. The steps can be embodied in machine-executable instructions, which can be used to cause a general-purpose or special-purpose processor programmed with the instructions to perform the steps. Alternatively, the steps of the present invention might be performed by specific hardware components that contain hardwired logic for performing the steps, or by any combination of programmed computer components and custom hardware components.

I. System Overview

The present invention is included in a system, known as WebTV™, for providing a user with access to the Internet. A user of a WebTV™ client generally accesses a WebTV™ server via a direct-dial telephone (POTS, for "plain old

4

telephone service"), ISDN (Integrated Services Digital Network), or other similar connection, in order to browse the Web, send and receive electronic mail (e-mail), and use various other WebTV™ network services. The WebTV™ network services are provided by WebTV™ servers using software residing within the WebTV™ servers in conjunction with software residing within a WebTV™ client.

FIG. 1 illustrates a basic configuration of the WebTV™ network according to one embodiment. A number of WebTV™ clients 1 are coupled to a modem pool 2 via direct-dial, bi-directional data connections 29, which may be telephone (POTS, i.e., "plain old telephone service"), ISDN (Integrated Services Digital Network), or any other similar type of connection. The modem pool 2 is coupled typically through a router, such as that conventionally known in the art, to a number of remote servers 4 via a conventional network infrastructure 3, such as the Internet. The WebTV™ system also includes a WebTV™ server 5, which specifically supports the WebTV™ clients 1. The WebTV™ clients 1 each have a connection to the WebTV™ server 5 either directly or through the modem pool 2 and the Internet 3. Note that the modem pool 2 is a conventional modem pool, such as those found today throughout the world providing access to the Internet and private networks.

Note that in this description, in order to facilitate explanation the WebTV™ server 5 is generally discussed as if it were a single device, and functions provided by the WebTV™ services are generally discussed as being performed by such single device. However, the WebTV™ server 5 may actually comprise multiple physical and logical devices connected in a distributed architecture, and the various functions discussed below which are provided by the WebTV™ services may actually be distributed among multiple WebTV™ server devices.

II. Client System

FIG. 2 illustrates a WebTV™ client 1. The WebTV™ client 1 includes an electronics unit 10 (hereinafter referred to as "the WebTV™ box 10"), an ordinary television set 12, and a remote control 11. In an alternative embodiment of the present invention, the WebTV™ box 10 is built into the television set 12 as an integral unit. The WebTV™ box 10 includes hardware and software for providing the user with a graphical user interface, by which the user can access the WebTV™ network services, browse the Web, send e-mail, and otherwise access the Internet.

The WebTV™ client 1 uses the television set 12 as a display device. The WebTV™ box 10 is coupled to the television set 12 by a video link 6. The video link 6 is an RF (radio frequency), S-video, composite video, or other equivalent form of video link. In the preferred embodiment, the client 1 includes both a standard modem and an ISDN modem, such that the communication link 29 between the WebTV™ box 10 and the server 5 can be either a telephone (POTS) connection 29a or an ISDN connection 29b. The WebTV™ box 10 receives power through a power line 7.

Remote control 11 is operated by the user in order to control the WebTV™ client 1 in browsing the Web, sending e-mail, and performing other Internet-related functions. The WebTV™ box 10 receives commands from remote control 11 via an infrared (IR) communication link. In alternative embodiments, the link between the remote control 11 and the WebTV™ box 10 may be RF or any equivalent mode of transmission.

III. Server System

The WebTV™ server 5 generally includes one or more computer systems generally having the architecture illus-

5

trated in FIG. 3. It should be noted that the illustrated architecture is only exemplary, the present invention is not constrained to this particular architecture. The illustrated architecture includes a central processing unit (CPU) 50, random access memory (RAM) 51, read-only memory (ROM) 52, a mass storage device 53, a modem 54, a network interface card (NIC) 55, and various other input/output (I/O) devices 56. Mass storage device 53 includes a magnetic, optical, or other equivalent storage medium. I/O devices 56 may include any or all of devices such as a display monitor, keyboard, cursor control device, etc. Modem 54 is used to communicate data to and from remote servers 4 via the Internet.

As noted above, the WebTV™ server 5 may actually comprise multiple physical and logical devices connected in a distributed architecture. Accordingly, NIC 55 is used to provide data communication with other devices that are part of the WebTV™ services. Modem 54 may also be used to communicate with other devices that are part of the WebTV™ services and which are not located in close geographic proximity to the illustrated device.

According to the present invention, the WebTV™ server 5 acts as a proxy in providing the WebTV™ client 1 with access to the Web and other WebTV™ services. More specifically, WebTV™ server 5 functions as a "caching proxy." FIG. 4A illustrates the caching feature of the WebTV™ server 5. In FIG. 4A, the WebTV™ server 5 is functionally located between the WebTV™ client 1 and the Internet infrastructure 3. The WebTV™ server 5 includes a proxy cache 65 which is functionally coupled to the WebTV™ client 1. The proxy cache 65 is used for temporary storage of Web documents, images, and other information which is frequently used by either the WebTV™ client 1 or the WebTV™ server 5.

A document transcoder 66 is functionally coupled between the proxy cache 65 and the Internet infrastructure 3. The document transcoder 66 includes software which is used to automatically revise the code of Web documents retrieved from the remote servers 4, for purposes which are described below.

The WebTV™ service provides a document database 61 and a user database 62, as illustrated in FIG. 4B. The user database 62 contains information that is used to control certain features relating to access privileges and capabilities of the user of the client 1. This information is used to regulate initial access to the WebTV™ service, as well as to regulate access to the individual services provided by the WebTV™ system, as will be described below. The document database 61 is a persistent database which stores certain diagnostic and historical information about each document and image retrieved by the server 5, as is now described.

A. Document Database

The basic purpose of the document database 61 is that, after a document has once been retrieved by the server 5, the stored information can be used by the server 5 to speed up processing and downloading of that document in response to all future requests for that document. In addition, the transcoding functions and various other functions of the WebTV™ service are facilitated by making use of the information stored in the document database 61, as will be described below.

Referring now to FIG. 5, the server 5 initially receives a document request from a client 1 (step 501). The document request will generally result from the user of the client 1

6

activating a hypertext anchor (link) on a Web page. The act of activating a hypertext anchor may consist of clicking on underlined text in a displayed Web page using a mouse, for example. The document request will typically (but not always) include the URL (Uniform Resource Locator) or other address of the selected anchor. Upon receiving the document request, the server 5 optionally accesses the document database 61 to retrieve stored information relating to the requested document (step 502). It should be noted that the document database 61 is not necessarily accessed in every case. The information retrieved from the document database 61 is used by the server 5 for determining, among other things, how long a requested document has been cached and/or whether the document is still valid. The criteria for determining validity of the stored document are discussed below. The server 5 retrieves the document from the cache 65 if the stored document is valid; otherwise, the server 5 retrieves the document from the appropriate remote server 4 (step 503). The server 5 automatically transcodes the document as necessary based on the information stored in the document database 61 (step 503). The transcoding functions are discussed further below.

The document database 61 includes certain historical and diagnostic information for every Web page that is accessed at any time by a WebTV™ client 1. As is well known, a Web page may correspond to a document written in a language such as HTML (Hypertext Mark-Up Language), VRML (Virtual Reality Modelling Language), or another suitable language. Alternatively, a Web page may represent an image, or a document which references one or more images. According to the present invention, once a document or image is retrieved by the WebTV™ server 5 from a remote server 4 for the first time, detailed information on this document or image is stored permanently in the document database 61. More specifically, for every Web page that is retrieved from a remote server 4, any or all of the following data are stored in the document database 61:

- 1) information identifying bugs (errors) or quirks in the Web page, or undesirable effects caused when the Web page is displayed by a client 1;
- 2) relevant bug-finding algorithms;
- 3) the date and time the Web page was last retrieved;
- 4) the date and time the Web page was most recently altered by the author;
- 5) a checksum for determining whether the Web page has been altered;
- 6) the size of the Web page (in terms of memory);
- 7) the type of Web page (e.g., HTML document, image, etc.);
- 8) a list of hypertext anchors (links) in the Web page and corresponding URLs;
- 9) a list of the most popular anchors based on the number of "hits" (requests from a client 1);
- 10) a list of related Web pages which can be prefetched;
- 11) whether the Web page has been redirected to another remote server 4;
- 12) a redirect address (if appropriate);
- 13) whether the redirect (if any) is temporary or permanent, and if permanent, the duration of the redirect;
- 14) if the Web page is an image, the size of the image in terms of both physical dimensions and memory space;
- 15) the sizes of in-line images (images displayed in text) referenced by the document defining the Web page;

7

- 16) the size of the largest image referenced by the document;
- 17) information identifying any image maps in the Web page;
- 18) whether to resize any images corresponding to the Web page;
- 19) an indication of any forms or tables in the Web page;
- 20) any unknown protocols;
- 21) any links to "dead" Web pages (i.e., pages which are no longer active);
- 22) the latency and throughput of the remote server 4 on which the Web page is located;
- 23) the character set of the document;
- 24) the vendor of the remote server 4 on which the Web page is located;
- 25) the geographic location of the remote server 4 on which the Web page is located;
- 26) the number of other Web pages which reference the subject Web page;
- 27) the compression algorithm used by the image or document;
- 28) the compression algorithm chosen by the transcoder;
- 29) a value indicating the popularity of the Web page based on the number of hits by clients; and
- 30) a value indicating the popularity of other Web pages which reference the subject Web page.

B. Transcoding

As mentioned above, the WebTV™ services provide a transcoder 66, which is used to rewrite certain portions of the code in an HTML document for various purposes. These purposes include: (1) correcting bugs in documents; (2) correcting undesirable effects which occur when a document is displayed by the client 1; (3) improving the efficiency of transmission of documents from the server 5 to the client 1; (4) matching hardware decompression technology within the client 1; (5) resizing images to fit on the television set 12; (6) converting documents into other formats to provide compatibility; (7) reducing latency experienced by a client 1 when displaying a Web page with in-line images (images displayed in text); and, (8) altering documents to fit into smaller memory spaces.

There are three transcoding modes used by the transcoder 66: (1) streaming, (2) buffered, and (3) deferred. Streaming transcoding refers to the transcoding of documents on a line-by-line basis as they are retrieved from a remote server 4 and downloaded to the client 1 (i.e., transcoding "on the fly"). Some documents, however, must first be buffered in the WebTV™ server 5 before transcoding and downloading them to the client 1. A document may need to be buffered before transmitting it to the client 1 if the type of changes to be made can only be made after the entire document has been retrieved from the remote server 4. Because the process of retrieving and downloading a document to the client 1 increases latency and decreases throughput, it is not desirable to buffer all documents. Therefore, the transcoder 66 accesses and uses information in the document database 61 relating to the requested document to first determine whether a requested document must be buffered for purposes of transcoding, before the document is retrieved from the remote server 4.

In the deferred mode, transcoding is deferred until after a requested document has been downloaded to a client 1. The deferred mode therefore reduces latency experienced by the

8

client 1 in receiving the document. Transcoding may be performed immediately after downloading or any time thereafter. For example, it may be convenient to perform transcoding during periods of low usage of WebTV™ services, such as at night. This mode is useful for certain types of transcoding which are not mandatory.

1. Transcoding for Bugs and Quirks

One characteristic of some prior art Web browsers is that they may experience failures ("crashes") because of bugs or unexpected features ("quirks") that are present in a Web document. Alternatively, quirks in a document may cause an undesirable result, even though the client does not crash. Therefore, the transcoding feature of the present invention provides a means for correcting certain bugs and quirks in a Web document. To be corrected by the transcoder 66, bugs and quirks must be identifiable by software running on the server 5. Consequently, the transcoder 66 will generally only correct conditions which have been previously discovered, such as those discovered during testing or reported by users. Once a bug or quirk is discovered, however, algorithms are added to the transcoder 66 to both detect the bug or quirk in the future in any Web document and to automatically correct it.

There are countless possibilities of bugs or quirks which might be encountered in a Web document. Therefore, no attempt will be made herein to provide an exhaustive list. Nonetheless, some examples may be useful at this point. Consider, for example, an HTML document that is downloaded from a remote server 4 and which contains a table having a width specified in the document as "0." This condition might cause a failure if the client were to attempt to display the document as written. This situation therefore, can be detected and corrected by the transcoder 66. Another example is a quirk in the document which causes quotations to be terminated with too many quotation marks. Once the quirk is first detected and an algorithm is written to recognize it, the transcoder 66 can automatically correct the quirk in any document.

If a given Web document has previously been retrieved by the server 5, there will be information regarding that document available in the document database 61 as described above. The information regarding this document will include whether or not the document included any bugs or quirks that required transcoding when the document was previously retrieved. The transcoder 66 utilizes this information to determine whether (1) the document is free of bugs and quirks, (2) the document has bugs or quirks which can be remedied by transcoding on the fly, or (3) the document has bugs or quirks which cannot be corrected on the fly (i.e., buffering is required).

FIG. 6 illustrates a routine for transcoding a Web document for purposes of eliminating bugs and quirks. Initially, the server 5 receives a document request from the client 1 (step 601). Next, the document database 61 is accessed to determine whether or not the requested document has been previously retrieved (step 602). If the document has not been previously retrieved, then the server 5 retrieves the document from the remote server 4 (step 609). Next, the retrieved document is analyzed for the presence of bugs or unusual conditions (step 610). Various diagnostic information is then stored in the document database 61 as a result of the analysis to note any bugs or quirks that were found (step 611). If any bugs or quirks were found which can be corrected by the transcoder 66, the document is then transcoded and saved to the proxy cache 65 (step 612). The transcoded document is

9

then downloaded to the client 1 (step 613). It should be noted that transcoding can be deferred until after the document has been downloaded, as described above; hence, the sequence of FIG. 6 is illustrative only.

If (in step 602) the requested document had been previously retrieved, then it is determined whether the requested document is still valid (step 603) and whether the document is present in the proxy cache 65 (step 604). If the document is no longer valid, then the document is retrieved from the remote server 4, analyzed for bugs and quirks, transcoded as required, and then downloaded to the client 1 as described above (steps 609-613). Methods for determining validity of a document are discussed below. If the document is still valid (step 603) and the document is present in the cache 65, the document is downloaded to the client 1 in its current form (as it is stored in the cache), since it has already been transcoded (step 608).

The document, however, may be valid but not present in the cache. This may be the case, for example, if the document has not been requested recently and the cache 65 has become too full to retain the requested document. In that case, the document is retrieved again from the remote server 4 (step 605) and then transcoded on the basis of the previously-acquired diagnostic information stored within the database 61 for that document. The document is then saved to the cache 65 (step 606). Note that because the document is still valid, it is assumed that the diagnostic information stored in the document database 61 for that document is still valid and that the transcoding can be performed on the basis of that information. Accordingly, once the document is transcoded, the transcoded document is downloaded to the client 1 (step 607). Again, note that transcoding can be deferred until after the document has been downloaded in some cases.

The validity of the requested document can be determined based on various different criteria. For example, some HTML documents specify a date on which the document was created, a length of time for which the document will be valid, or both. The validity determination can be based upon such information. For example, a document which specifies only the date of creation can be automatically deemed invalid after a predetermined period of time has passed.

Alternatively, validity can be based upon the popularity of the requested document. "Popularity" can be quantified based upon the number of hits for that document, which is tracked in the document database 61. For example, it might be prudent to simply assign a relatively short period of validity to a document which is very popular and a longer period of validity to a document which is less popular.

Another alternative basis for the validity of a document is the observed rate of change of the document. Again, data in the persistent document database 61 can be used. That is, because the document database 61 stores the date and time on which the document was last observed to change, the server 5 can approximate how often the document actually changes. A document or image which is observed to change frequently (e.g., a weather map or a news page) can be assigned a relatively short period of validity. It will be recognized that numerous other ways of determining validity are possible.

2. Transcoding to Reduce Latency

Another purpose for transcoding is to allow documents requested by a client 1 to be displayed by the client 1 more rapidly. Many HTML documents contain references to "in-line" images, or images that will be displayed in text in a

10

Web page. The normal process used in the prior art to display a Web page having in-line images is that the HTML document referencing the image is first downloaded to the client, followed by the client's requesting the referenced image. The referenced image is then retrieved from the remote server on which it is located and downloaded to the client. One problem associated with the prior art, however, is that the speed with which a complete Web page can be displayed to the user is often limited by the time it takes to retrieve in-line images. One reason for this is that it simply takes time to retrieve the image itself after the referencing document has been retrieved. Another reason is that, in the prior art, if the referencing document does not specify the size of the image, the Web page generally cannot be displayed until the image itself has been retrieved. The present invention overcomes these limitations.

According to the present invention, information stored in the document database 61 regarding the in-line images is used to transcode the referencing document in order to reduce latency in displaying the Web page. Once any document which references an in-line image is initially retrieved by the server 5, the fact that the document references an in-line image is stored in the document database 61. In addition, the size of the image is determined, either from the document (if specified) or from the image itself, and then stored in the document database 61. Consequently, for documents which do not specify the size of their in-line images, the size information stored in the database 61 is then used the next time the document is requested in order to reduce latency in downloading and displaying the Web page.

Refer now to FIG. 7, which illustrates a routine for reducing latency when downloading a document referencing an image to a client 1. Assume that a client 1 sends a request to the server 5 for an HTML document containing a reference to an in-line image. Assume further that the size of the image is not specified in the document itself. Initially, the server 5 determines whether that document has been previously retrieved (step 701). If not, the standard initial retrieval and transcoding procedure is followed (step 706), as described in connection with FIG. 6. If, however, the document has been previously retrieved, then the transcoder 66 accesses the size information stored in the document database 61 for the in-line image (step 702). Based on this size information, the HTML document is transcoded such that, when the Web page is initially displayed by the client 1, the area in which the image belongs is replaced by a blank region enveloping the shape of the image (step 703). Thus, any in-line image referenced by a document is displayed initially as a blank region. Consequently, the client 1 can immediately display the Web page corresponding to the HTML document even before the referenced image has been retrieved or downloaded (i.e., even before the size of the image is known to the client 1).

As the transcoded HTML document is downloaded to the client, the image is retrieved from the appropriate remote server 4 (step 704). Once the image is retrieved from the remote server 4 and downloaded to the client 1, the client 1 replaces the blank area in the Web page with the actual image (step 705).

3. Transcoding to Display Web Pages on a Television

As noted above, the client 1 utilizes an ordinary television set 12 as a display device. However, images in Web pages are generally formatted for display on a computer monitor, not a television set. Consequently, the transcoding function

11

of the present invention is used to resize images for display on the television set 12. This includes resealing images as necessary to avoid truncation when displayed on the television set 12.

It should be noted that prior art Web browsers which operate on computer monitors typically use resizable windows. Hence, the size of the visible region varies from client to client. However, because the web browser used by the WebTV™ client 1 is specifically designed for display on a television set, the present invention allows documents and images to be formatted when they are cached.

4. Transcoding for Transmission Efficiency

Documents retrieved by the server 5 are also transcoded to improve transmission efficiency. In particular, documents can be transcoded in order to reduce high frequency components in order to reduce interface flicker when they are displayed on a television set. Various methods for coding software or hardware to reduce perceptual interface flicker are described in co-pending U.S. patent application Ser. No. 08/656,923, filed on Jun. 3, 1996.

Documents can also be transcoded in order to lower the resolution of the displayed Web page. Reducing the resolution is desirable, because images formatted for computer systems will generally have a higher resolution than the NTSC (National Television Standards Committee) video format used by conventional television sets. Since the NTSC video does not have the bandwidth to reproduce the resolution of computer-formatted images, the bandwidth consumed in transmitting images to the client 1 at such a high resolution would be wasted.

5. Other Uses for Transcoding

Transcoding is also used by the present invention to recode a document using new formats into older, compatible formats. Images are often displayed in the JPEG (Joint Picture Experts Group) format or the GIF image format. JPEG often consumes less bandwidth than GIF, however. Consequently, images which are retrieved in GIF format are sometimes transcoded into JPEG format. Methods for generally converting images between GIF and JPEG formats are well known.

Other uses for transcoding include transcoding audio files. For example, audio may be transcoded into different formats in order to achieve a desired balance between memory usage, sound quality, and data transfer rate. In addition, audio may be transcoded from a file format (e.g., an "AU" file) to a streaming format (e.g., MPEG 1 audio). Yet another use of audio transcoding is the transcoding of MIDI (Musical Instrument Digital Interface) data to streaming variants of MIDI.

Additionally, documents or images requiring a large amount of memory (e.g., long lists) can be transcoded in order to consume less memory space in the client 1. This may involve, for example, separating a large document or image into multiple sections. For example, the server 5 can insert tags at appropriate locations in the original document so that the document appears to the client 1 as multiple Web pages. Hence, while viewing a given page representing a portion of the original document, the user can view the next page (i.e., the next portion of the original document) by activating a button on the screen as if it were an ordinary hypertext anchor.

C. Proxying

As noted above, the server 5 functions as a proxy on behalf of the client 1 for purposes of accessing the Web. The

12

document database 61 is used in various ways to facilitate this proxy role, as will now be described.

1. Updating Cached Documents

It is desirable to store frequently-requested HTML documents and images in the proxy cache 65 to further reduce latency in providing Web pages to the client 1. However, because some documents and images change over time, documents in the cache 65 will not be valid indefinitely, as mentioned above. A weather map or a news-related Web page, for example, are likely to be updated quite frequently. Consequently, it is desirable for the server 5 to have the ability to estimate the frequency with which documents change, in order to determine how long a document can safely remain within the proxy cache 65 without being updated.

The persistent database 65 is used to store the date and time of the last several fetches of each document and image retrieved from a remote server 4, along with an indication of any changes that were detected, if any. A document or image which has been stored in the cache 65 is then retrieved on a periodic basis to determine if it has been changed. Change status information indicating whether the document has changed since the previous fetch is then stored in the document database 61. If no changes are detected, then the time interval between fetches of this document is increased. If the document has changed, the time interval is maintained or decreased. As a result, items in the cache 65 which change frequently will be automatically updated at frequent intervals, whereas documents which do not change often will be replaced in the cache less frequently.

FIG. 8 illustrates a routine for updating documents stored in the proxy cache 65 using data stored in the document database 61. Assume a document X has been stored in the proxy cache 65. Document X remains in the cache 65 until a predetermined update period T_1 expires (step 801). Upon the expiration of the update period T_1 , the document X is again retrieved from the appropriate remote server 4 (step 802). The newly-retrieved document X is then compared to the cached version of document X (step 803). If the document has changed, then the cached version of document X is replaced with the newly-retrieved version of document X (step 806). If not, then the update period T_1 is increased according to a predetermined time increment ΔT_1 (step 804). In any case, the date and time and the change status of document X is saved to the document database 61 (step 805).

2. Document and Image Prefetching

The document database 61 is also used by the server 5 to store prefetching information relating to documents and images. In particular, the database stores, for each document that has been retrieved, a list of images referenced by the document, if any, and their locations. Consequently, the next time a document is requested by a client 1, the images can be immediately retrieved by the server 5 (from the cache 65, if available, or from the remote server 4), even before the client 1 requests them. This procedure improves the speed with which requested Web pages are downloaded to the client.

The document database 61 is also used to facilitate a process referred to as "server-advised client prefetching." Server-advised client prefetching allows the server 5 to inform the client 1 of documents or images which are popular to allow the client 1 to perform the prefetching. In particular, for any given document, a list is maintained in the

13

server 5 of the most popular hypertext anchors in that document (i.e., those which have previously received a large number of hits). When that document is requested by the client 1, the server 5 provides the client 1 with an indication of these popular links.

3. Redirects

Web pages are sometimes forwarded from the remote server on which they are initially placed to a different location. Under the HTTP (Hypertext Transport Protocol), such forwarding is sometimes referred to as a "redirect." When an HTML document is initially stored on one remote server and then later transferred to another remote server, the first remote server will provide, in response to a request for that document, an indication that the document has been transferred to a new remote server. This indication generally includes a forwarding address ("redirect address"), which is generally a URL.

In the prior art, when a computer requesting a Web page receives a redirect, it must then submit a new request to the redirect address. Having to submit a second request and wait for a second response consumes time and increases overall latency. Consequently, the present invention uses the document database 61 to store any redirect address for each document or image. Any time a redirected document is requested, the server 5 automatically accesses the redirect address to retrieve the document. The document or image is provided to the client 1 based on only a single request from the client 1. The change in location of the redirected document or image remains completely transparent to the client 1.

FIG. 9 illustrates a routine performed by the server 5 in accessing documents which may have been forwarded to a new remote server. Initially, the server 5 receives a request for a document, which generally includes an address (step 901). The server 5 then accesses the document database 61 to determine whether there is a redirect address for the requested document (step 902). If there is no redirect address, then the server 5 accesses a remote server 4 based on the address provided in the document request from the client 1 (step 903). Assuming that the remote server 4 does not respond to the server 5 with a redirect (step 904), the document is retrieved and downloaded to the client 1 by the server 5 (step 907). If, however, a redirect address was stored in the document database 61 (step 902), then the server 5 accesses the requested document according to the redirect address (step 906). Or, if the remote server 4 responded with a redirect (step 904), then the server 5 saves the redirect address to the document database 61 (step 905) and accesses the requested document according to the redirect address (step 906).

4. Other Proxy Functions

The document database 61 also stores information relating to the performance of each remote server 4 from which a document is retrieved. This information includes the latency and throughput of the remote server 4. Such information can be valuable in instances where a remote server 4 has a history of responding slowly. For example, when the document is requested, this knowledge can be used by the server 5 to provide a predefined signal to the client 1. The client 1 can, in response to the signal, indicate to the user that a delay is likely and give the user the option of canceling the request.

5. Backoff Mode

Although the server 5 generally operates in the proxy mode, it can also enter a "backoff mode" in which the server

14

5 does not act as a proxy, or the server 5 performs only certain aspects of the normal proxying functions. For example, if the proxy cache 65 is overloaded, then the server 5 can enter a backoff mode in which documents are not cached but are transcoded as required. Alternatively, during times when the server 5 is severely overloaded with network traffic, the server 5 may instruct the client 1 to bypass the server 5 and contact remote servers 4 directly for a specified time or until further notice. Or, the server 5 can enter a flexible backoff mode in which the client 1 will be instructed to contact a remote server 4 directly only for certain Web sites for a limited period of time.

D. Access to WebTV™ Services

The WebTV™ server 5 provides various services to the client 1, such as proxying and electronic mail ("e-mail"). In the prior art, certain difficulties are associated with allowing a client computer access to different services of an Internet service, as will now be explained with reference to FIG. 10.

FIG. 10 illustrates a client-server system according to one prior art embodiment. The server 76 provides various services A, B, and C. The server 76 includes a database 71 for storing information on the user's access privileges to services A, B, and C. The client 75 of the embodiment of FIG. 10 accesses any of services A, B, and C by contacting that service directly. The contacted service then accesses the database 71, which stores the access privileges of the client 75, to determine whether the client 75 should be allowed to access that service. Hence, each service provided by the server 76 requires direct access to the database 71. This architecture results in a large number of accesses being made to the database 71, which is undesirable. In addition, the fact that each service independently has access to the database 71 raises security concerns. Specifically, it can be difficult to isolate sensitive user information. The present invention overcomes such difficulties using a technique which is now described.

1. Tickets Containing Privileges And Capabilities

As shown in FIG. 11, the server 5 provides a number of services D, E, and F 77, 79, and 80, respectively, and a log-in service 78. The log-in service 78 is used specifically to control initial log-on procedures by a client 1. The log-in service 78 has exclusive access to the user database 62 (discussed above with respect to FIG. 4B). The log-in service 78 and the user database 62 are located within a first security zone 84. Service D is located within a second security zone 86, while services E and F are contained within a third security zone 88. Note that the specific arrangement of security zones 84, 86, and 88 with respect to services D, E, and F is illustrative only.

The user database 62 of the present invention stores various information pertaining to each authorized user of a client 1. This information includes account information, a list of the WebTV™ services that are available to the particular user, and certain user preferences. For example, a particular user may not wish his client 1 to be used to access Web pages having adult-oriented subject matter. Consequently, the user would request that his account be filtered to prevent access to such material. This request would then be stored as part of the user data in the user database 62.

With regard to user preferences, the hypertext links selected by a given user can be tracked, and those having the largest number can be stored in the user database 62. The list can then be provided to the client 1 for use in generating a

15

menu screen of the user's favorite Web sites, to allow the user to directly access those Web sites. The list can also be used by the server 5 to analyze the user's interests and to formulate and provide to the user a list of new Web sites which the user is likely to be interested in. The list might be composed by associated key words in Web pages selected by the user with other Web pages.

Referring again to FIG. 11, in response to a log-on request by a client 1, the log-in service 78 consults the user database 62 to determine if access to the server 5 by this particular client 1 is authorized. Assuming access is authorized, the log-in service 78 retrieves certain user information pertaining to this particular client 1 from the user database 62. The log-in service then generates a "ticket" 82, which is an information packet including the retrieved information. The ticket 82 is then provided to the client 1 which requested access.

The ticket 82 includes all information necessary to describe the access privileges of a particular user with respect to all services provided by the server 5. For example, the ticket may include the user name registered to the client 1, the e-mail address assigned to client 1, and any filtering requested by the user with respect to viewing Web sites. Each time the user requests access to one of the services D, E, or F, the client 1 submits a copy of the ticket 82 to that service. The requested service can then determine from the copy of the ticket 82 whether access to that service by that client 1 is authorized and, if so, any important information relating to such access.

None of the services provided by the server 5, other than the log-in service 78, has access to the user database 62. Hence, any security-sensitive information can be isolated within the user database 62 and the log-in service 78. Such isolation allows the individual services provided by the server 5 to be placed within separate "firewalls" (security regions), illustrated as security zones 84, 86, and 88. In addition, this technique greatly reduces the number of accesses required to the user database 62 compared to the prior art embodiment illustrated in FIG. 10.

2. Redundancy of Services and Load Balancing

The present invention also includes certain redundancies in the various services provided by the server 5. In particular, a given service (e.g., e-mail) can be provided by more than one physical or logical device. Each such device is considered a "provider" of that service. If a given provider is overloaded, or if the client 1 is unable to contact that provider, the client 1 can contact any of the other providers of that service. When the server 5 receives a log-in request from a client 1, in addition to generating the above-described ticket 82, the log-in service 78 dynamically generates a list of available WebTV™ services and provides this list to the client 1.

The server 5 can update the list of services used by any client 1 to reflect services becoming unavailable or services coming on-line. Also, the list of services provided to each client 1 can be updated by the server 5 based upon changes in the loading of the server 5, in order to optimize traffic on the server 5. In addition, a client's list of services can be updated by services other than the log-in service 78, such that one service can effectively introduce another service to the client 1. For example, the e-mail service may provide a client 1 with the name, port number and IP of its address book service. Thus, one service can effectively, and securely within the same chain of trust, introduce another service to the client 1.

16

This list of services includes the name of each service, a port number for the provider of each service, and an IP (Internet Protocol) for each service. Different providers of the same service are designated by the same name, but different port numbers and/or IPs. Note that in a standard URL, the protocol is normally specified at the beginning of the URL, such as "HTTP://www. . . ." under the HTTP protocol. However, according to the present invention, the normal protocol designation (i.e., "HTTP") in the URL is replaced with the name of the service, since the port number and IP for each service are known to the client 1. Hence, the client 1 can access any of the redundant providers of a given service using the same URL. This procedure effectively adds a level of indirection to all accesses made to any WebTV™ service and automatically adds redundancy to the proxy service. It should also be noted that separate service names can also refer to the same service.

Assume, for example, that the e-mail service provided by the WebTV™ system is designated by the service name "WTV-mailto." A client 1 can access any provider of this e-mail service using the same URL. The client 1 merely chooses the appropriate port number and IP number to distinguish between providers. If the client 1 is unable to connect to one e-mail provider, it can simply contact the next one in the list.

Thus, at log-in time, a client 1 is provided with both a ticket containing privileges and capabilities as well as a list of service providers, as illustrated in FIG. 12. Initially, the log-in service 78 determines whether the user of client 1 is a valid user (step 1201). If not, log-in is denied (step 1205). If the user is a valid user, then the log-in service 78 gathers user information from the user database 62 and generates a ticket 82 (step 1202). The log-in service 78 also generates the above-described list of services (step 1203). The ticket 82 and the list of services are then downloaded to the client 1 (step 1204).

3. Asynchronous Notification to Clients by Server

Another limitation associated with prior art Internet servers is the inability to provide asynchronous notification information to the client in the absence of a request from the client to do so. It would be desirable, for example, for a server to notify a client on its own initiative when a particular Web page has changed or that a particular service is inaccessible. The server 5 of the present invention provides such capability, and the client 1 is configured to receive and decode such notifications. For example, the client 1 can receive updates of its listing of service providers from the server 5 at various points in time, as already described. Similarly, if a particular service provider becomes unavailable, that fact will be automatically communicated to the client 1. As another example, if e-mail addressed to the user has been received by the server 5, then the server 5 will send a message to the client 1 indicating this fact. The client 1 will then notify the user that e-mail is waiting by a message displayed on the television set 12 or by an LED (light emitting diode) built into the housing of WebTV™ box 10.

Thus, a method and apparatus have been described for providing proxying and transcoding of documents in a network. The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

17

What is claimed and desired to be secured by United States Letters Patent is:

1. In a proxying server coupled to a client and to a remote server, the proxying server operating as a proxy on behalf of the client for accessing the remote server, a method of retrieving and transcoding a document requested by the client, the method comprising the steps of:

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document in order to perform at least one function selected from the following functions:

matching decompression requirements at the client; converting the document into a format compatible for the client;

reducing latency experienced by the client; and altering the document to fit into smaller memory space; and

transmitting the document to the client.

2. A method according to claim 1, wherein the request includes a URL (Uniform Resource Locator).

3. A method according to claim 1, wherein the step of transcoding is performed while the step of retrieving the document is performed.

4. A method according to claim 1, wherein the step of transcoding comprises the step of reading at least a portion of the document from a proxy cache located in the proxying server.

5. A method according to claim 1, wherein the step of transmitting the document to the client is performed prior to performing the step of transcoding, at the proxying server, the data in the document.

6. A method according to claim 1, wherein the step of transmitting the document to the client comprises the step of transmitting the transcoded document to the client, wherein the step of transmitting the document to the client is performed after the step of transcoding, at the proxying server, the data in the document.

7. A method according to claim 1, wherein the document includes a link to another document, the link including a retrieval address, and wherein the step of transcoding at the proxying server the data in the document comprises the step of updating the link

8. A method according to claim 7, wherein the other document is an image, and wherein the step of updating the link comprises the step of adding information to the document indicating the size of the image.

9. A method according to claim 8, wherein the other document is inaccessible to the proxying server, and wherein the step of updating the link comprises the step of removing the link.

10. A method according to claim 7, wherein the other document has been relocated from the retrieval address to a redirect address, and wherein the step of updating the link comprises the step of updating the link to correspond to the redirect address.

11. A method according to claim 1, wherein the client includes a television display, wherein the document references an image, and wherein the step of transcoding, at the proxying server, the data in the document is conducted in order to perform at least the function of converting the document into a compatible format for the client, and comprises the step of revising the data such that the image is sized for display on the television display.

18

12. A method according to claim 1, wherein the document references an image having a first image format, and wherein the step of transcoding, at the proxying server, the document is conducted in order to perform at least the function of converting the document into a compatible format for the client, and includes the step of converting the first image format to a second image format.

13. A method according to claim 1, further comprising the steps of:

identifying an image referenced by the document;

determining whether the image has been previously retrieved by the proxying server; and

if the image has been previously retrieved by the proxying server, accessing information stored in the proxying server indicating the size of the image;

wherein the step of transcoding, at the proxying server, the data in the document is conducted in order to perform at least the function of reducing latency experienced by the client, and comprises the step of using the information indicating the size of the image to revise the data of the document to allow the document to be partially displayed by the client before the image is received by the client.

14. A method according to claim 13, wherein the step of transcoding, at the proxying server, the data in the document comprises the following step:

if the image has been previously retrieved by the proxying server, transcoding at the proxying server the data in the document without the image but providing a space for the image to be later inserted; and

wherein the step of transmitting the document to the client comprises the following steps:

transmitting the document to the client without the image but providing the space for the image to be later inserted; and

after transmitting the document to the client without the image, transmitting the image to the client.

15. In a proxying server coupled to a client and to a remote server, the proxying server operating as a proxy on behalf of the client for accessing the remote server, a method of retrieving and transcoding a document requested by the client, the method comprising the steps of:

providing a persistent database at the proxying server, the persistent database including information relating to the document;

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document using the information included in the persistent database in order to perform at least one function selected from the following functions:

matching decompression requirements at the client; converting the document into a format compatible for the client;

reducing latency experienced by the client; and altering the document to fit into smaller memory space; and

transmitting the document to the client.

16. A method according to claim 15, further comprising the step of storing in the persistent database validity information corresponding to the document.

19

17. A method according to claim 16, wherein the step of retrieving the document comprises the following steps:

retrieving the document from the persistent database if the validity information corresponding to the document indicates that the document is valid; and

retrieving the document from the remote server if the validity information corresponding to the document indicates that the document is not valid.

18. A method according to claim 16, wherein the step of storing in the persistent database validity information corresponding to the document comprises the step of observing a rate of change of the document.

19. A method according to claim 18, wherein the step of observing a frequency of change of the document comprises the step of periodically retrieving the document, wherein successive retrievals of the document are separated in time by at least a time interval.

20. A method according to claim 19, wherein the step of periodically retrieving the document comprises the following steps:

retrieving the document at a first time;

storing the document retrieved at the first time in a memory;

retrieving the document at a second time, the second time being at least a time interval after the first time;

determining whether the document retrieved at the second time has changed compared to the document retrieved at the first time;

if the document retrieved at the second time is different than the document retrieved at the first time, replacing the document retrieved at the first time with the document retrieved at the second time in the memory; and

if the document retrieved at the second time is the same as the document retrieved at the first time, extending the time interval.

21. A method according to claim 15, further comprising the step of storing in the persistent database performance information relating to performance of the remote server when accessing the document.

22. A method according to claim 21, wherein the performance information comprises a latency value.

23. A method according to claim 15, further comprising the step of storing in the persistent database information for optimizing memory usage by the client.

24. A method according to claim 15, wherein the step of retrieving the document comprises the following steps:

determining whether a redirect address corresponding to the document is stored in the persistent database;

if the redirect address corresponding to the document is stored in the persistent database, accessing the remote server using the redirect address, the remote server corresponding to the redirect address; and

if the redirect address corresponding to the document is not stored in the persistent database, the method further comprises the following steps:

accessing a previous remote server at which the document previously resided;

obtaining the redirect address from the previous remote server;

storing the redirect address in the persistent database; and

20

accessing the remote server using the redirect address.

25. A computer program product for implementing, in a proxying server coupled to a client and to a remote server, a method of retrieving and transcoding a document requested by the client, the computer program product comprising a computer-readable medium carrying computer-executable instructions for causing the proxying server to perform acts included in the method, said acts comprising:

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document in order to perform at least one function selected from the following functions:

matching decompression requirements at the client;

converting the document into a format compatible for the client;

reducing latency experienced by the client; and
altering the document to fit into smaller memory space; and

transmitting the document to the client.

26. A computer program product according to claim 25, wherein the computer-executable instructions comprise a plurality of program code means for transcoding at least a portion of the document, including:

program code means for transcoding at least a portion of the document so as to match decompression requirements at the client;

program code means for transcoding at least a portion of the document so as to convert the document into a format compatible with the client;

program code means for transcoding at least a portion of the document so as to reduce latency experienced by the client; and

program code means for transcoding at least a portion of the document so as to alter the document to fit into smaller memory space.

27. A computer program product according to claim 26, wherein the act of transcoding comprises selecting at least one of said plurality of program code means for transcoding for use in the act of transcoding.

28. A computer program product according to claim 25, wherein the act of transcoding is performed while the act of retrieving the document is performed.

29. A computer program product according to claim 25, wherein the act of transcoding comprises reading at least a portion of the document from a proxy cache located in the proxying server.

30. A computer program product according to claim 25, wherein the act of transmitting the document to the client is performed prior to performing the act of transcoding.

31. A computer program product according to claim 25, wherein the act of transmitting the document to the client comprises the act of transmitting the transcoded document to the client, wherein the act of transmitting the document to the client is performed after the act of transcoding.

* * * * *



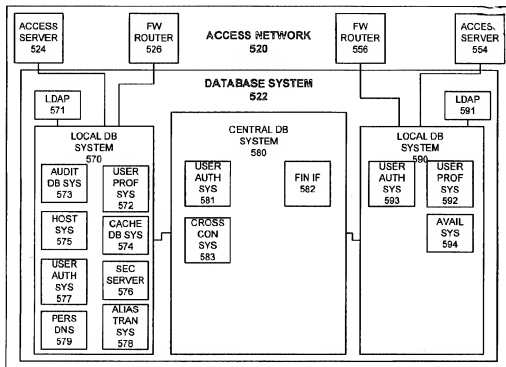
US006697806B1

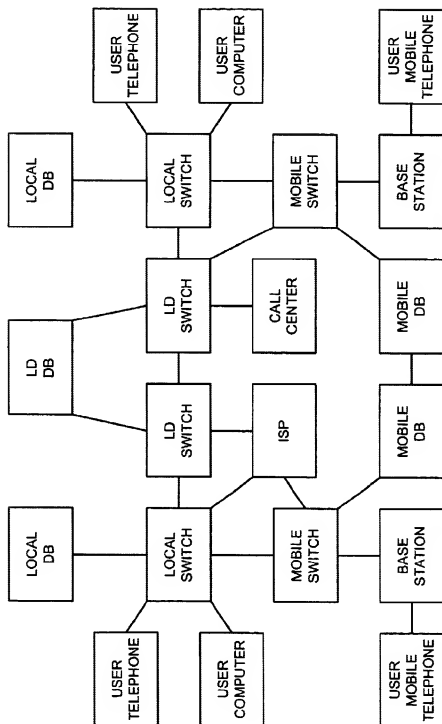
**(12) United States Patent
Cook****(10) Patent No.: US 6,697,806 B1
(45) Date of Patent: Feb. 24, 2004****(54) ACCESS NETWORK AUTHORIZATION****(75) Inventor: Fred S. Cook, Olathe, KS (US)****(73) Assignee: Sprint Communications Company, L.P., Overland Park, KS (US)****(*) Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.**(21) Appl. No.: 09/574,978****(22) Filed: May 19, 2000****Related U.S. Application Data****(63)** Continuation of application No. 09/556,276, filed on Apr. 24, 2000.**(51) Int. Cl.⁷ G06F 17/30****(52) U.S. Cl. 707/10; 707/3; 707/9;
709/227; 709/203; 709/219****(58) Field of Search 707/1-3, 9-10,
707/4, 104.1; 709/223-227, 219, 203****(56) References Cited****U.S. PATENT DOCUMENTS**5,884,312 A * 3/1999 Dustan et al. 707/10
6,151,601 A * 11/2000 Papierniak et al. 707/1

* cited by examiner

Primary Examiner—Jean R. Homere
Assistant Examiner—Mohammad Ali**(57) ABSTRACT**

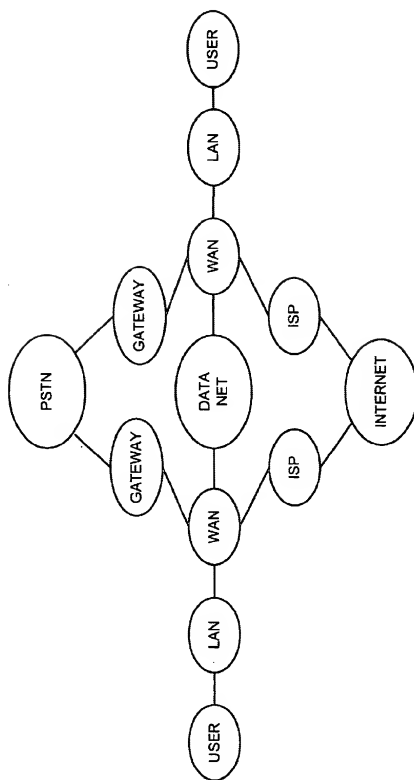
An access communication system provides access between a user system and a plurality of communication networks. The plurality of communication networks provide services to a user in the user system. An access communication system includes a local database system and an access server that is connected to the user system and the plurality of communication networks. The local database system receives a user logon. The local database system then processes the user logon to determine if the user is allowed access to the access communication system based on a local database system. The local database system then provides access to the access communication system to the user in response to the determination that the user is allowed access based on the local database system. The local database system then generates an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system. The local database system receives and processes an authorization response indicating whether the user is allowed to use the access system from the second database system. The local database system then provides access to the access communication system to the user in response to the authorization response that allows the user to use the access communication system.

24 Claims, 63 Drawing Sheets



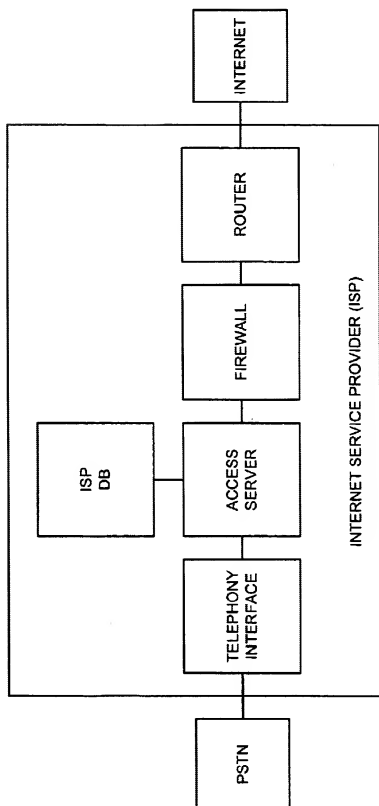
PRIOR ART

FIG. 1



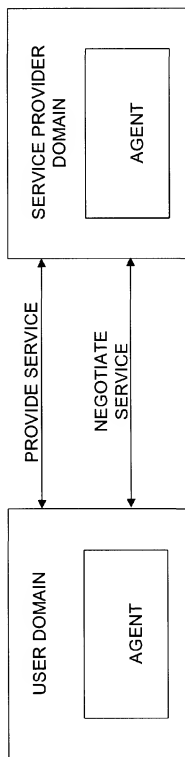
PRIOR ART

FIG. 2



PRIOR ART

FIG. 3



PRIOR ART

FIG. 4

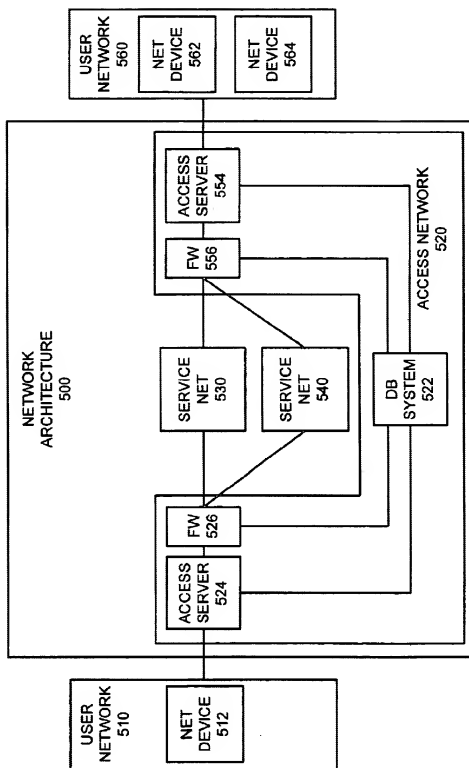


FIG. 5

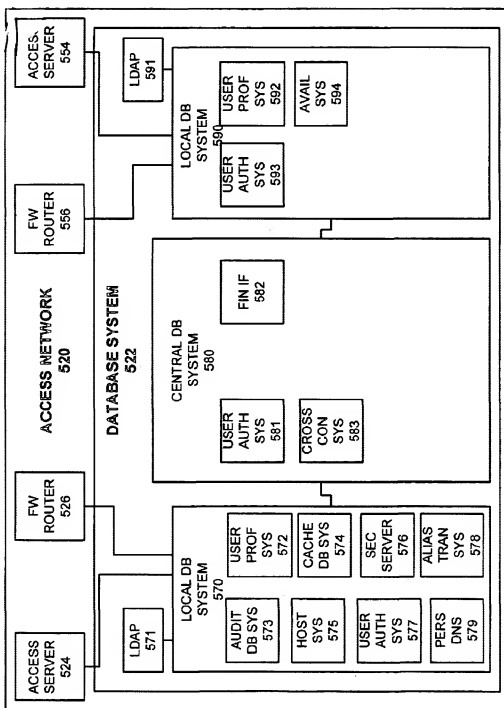


FIG. 6

USER ID	PASSWORD	NAME	ACCOUNT NUMBER	SERVICES	ADDRESS	BILLING CODE	CLASS	GROUP	SHELL	MACROS

FIG. 7

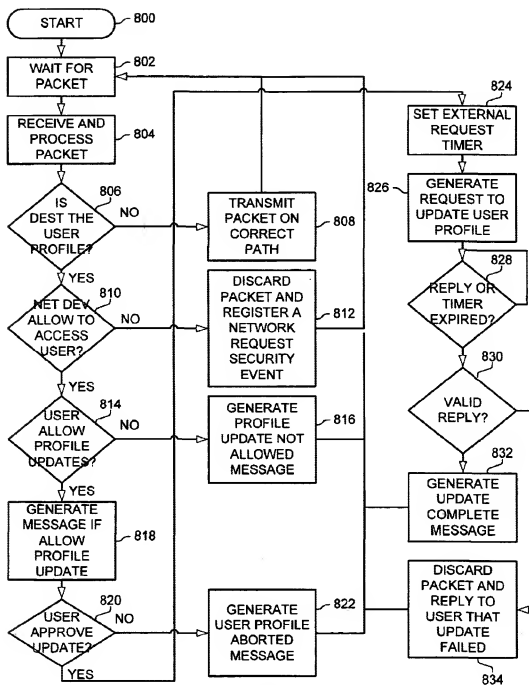


FIG. 8

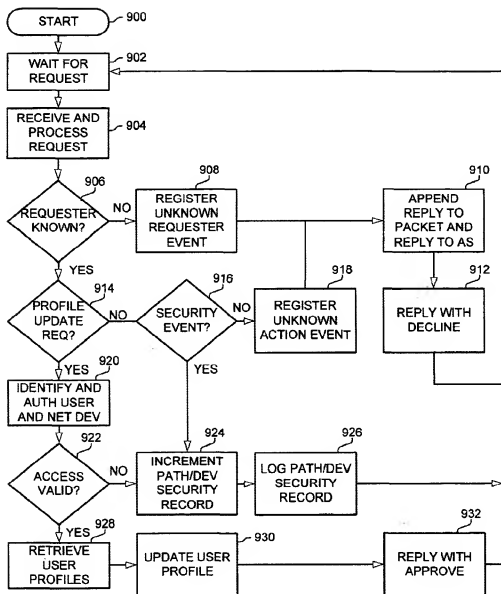


FIG. 9

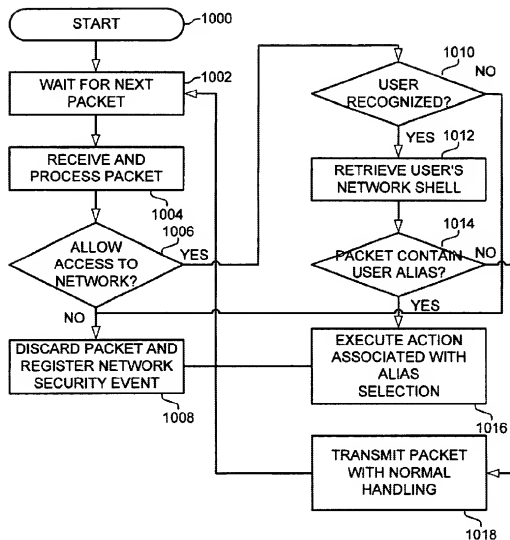


FIG. 10

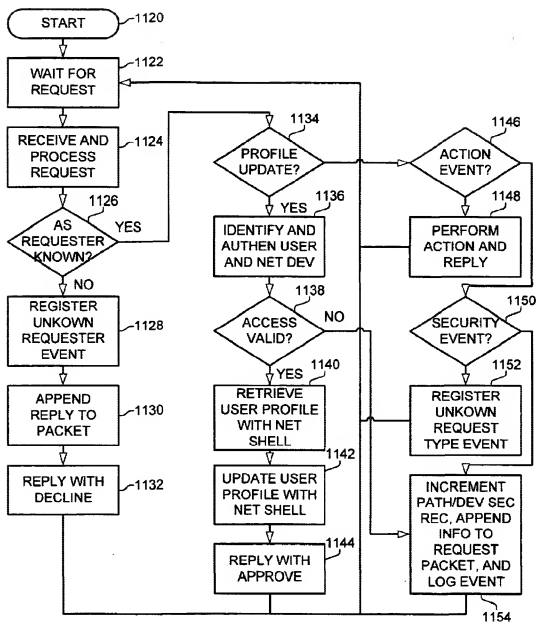


FIG. 11

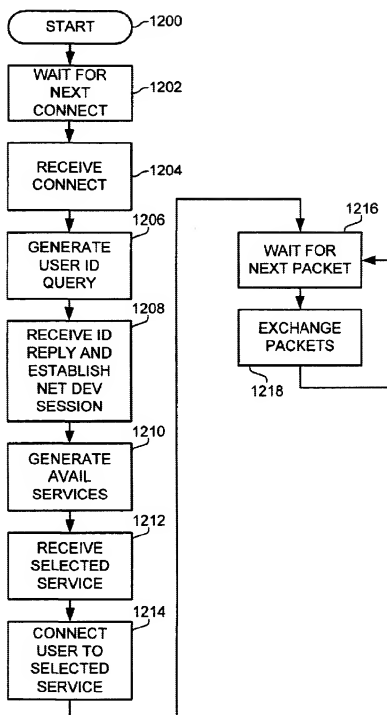


FIG. 12

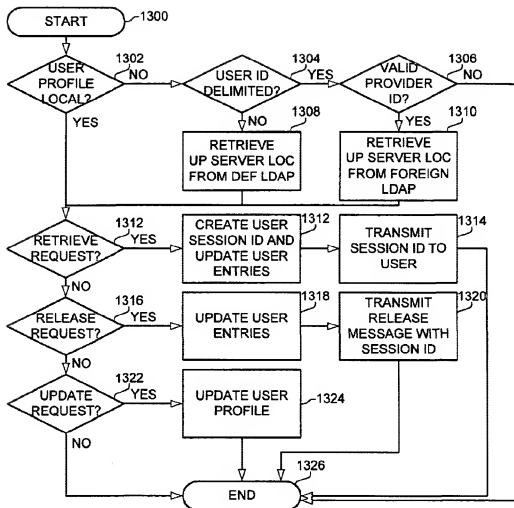


FIG. 13

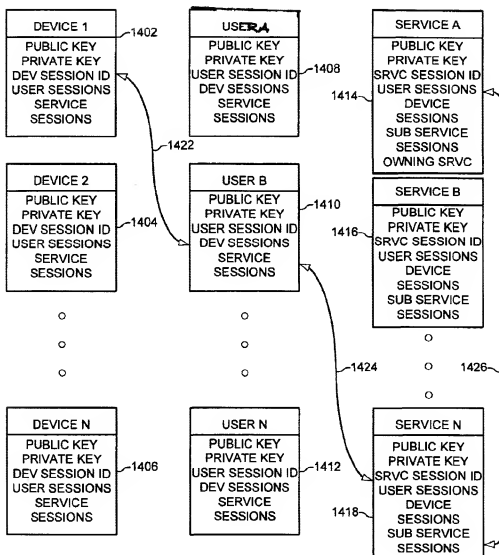
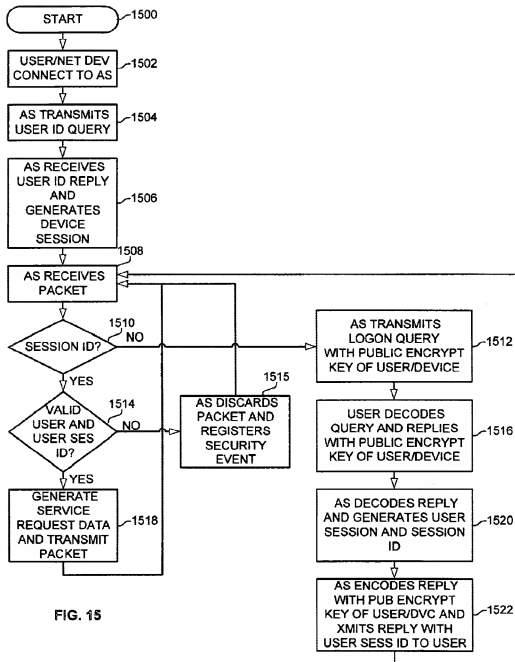


FIG. 14



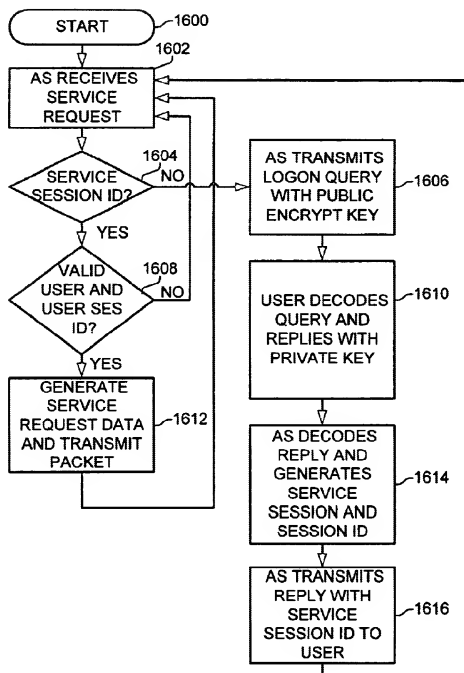


FIG. 16

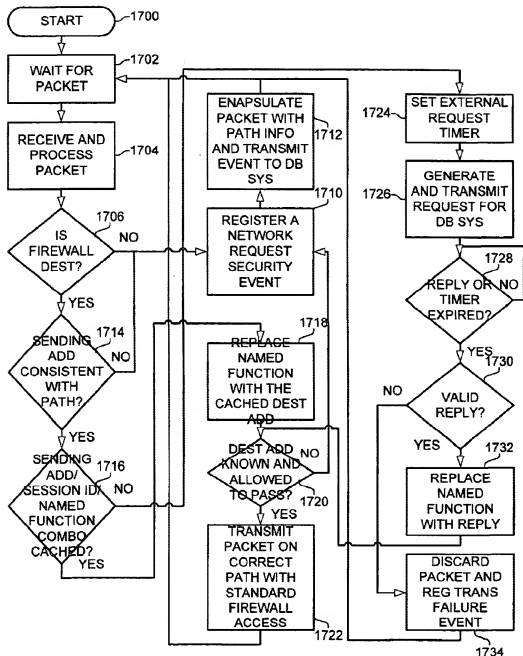


FIG. 17

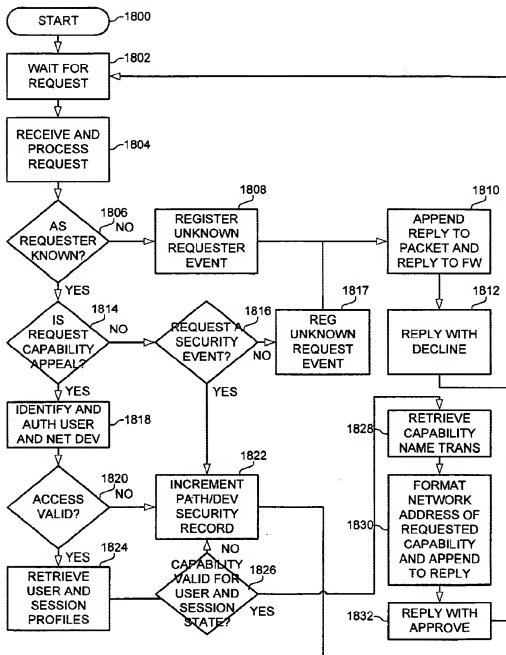


FIG. 18

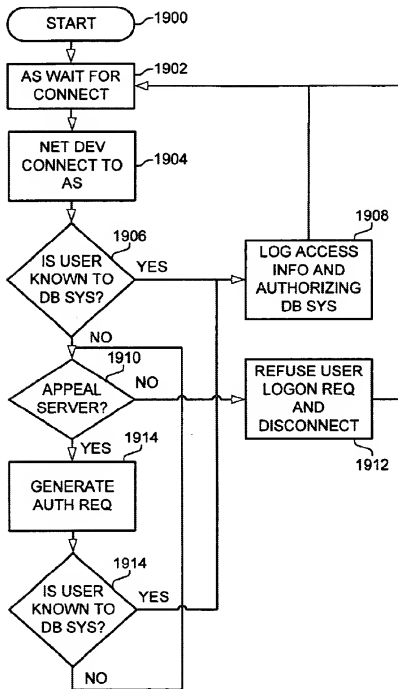


FIG. 19

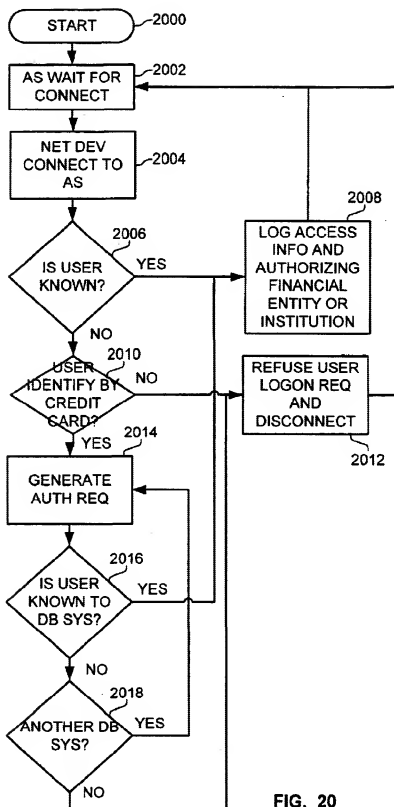


FIG. 20

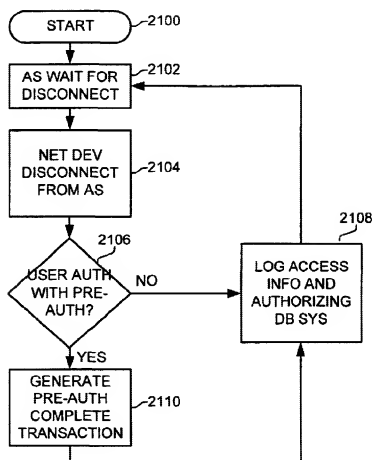


FIG. 21

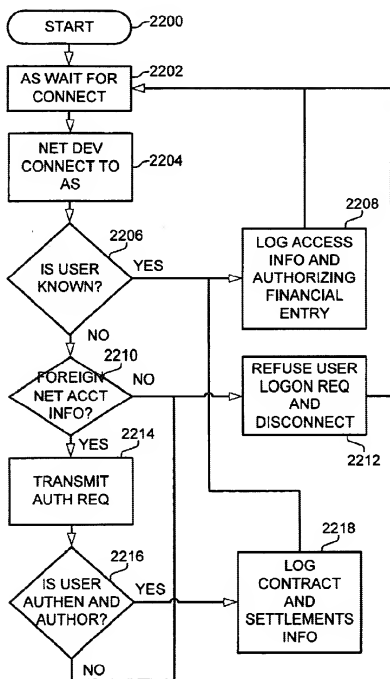
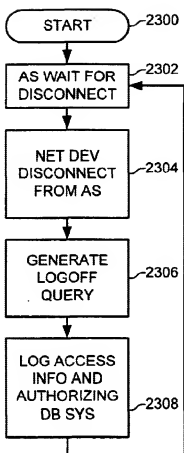


FIG. 22

**FIG. 23**

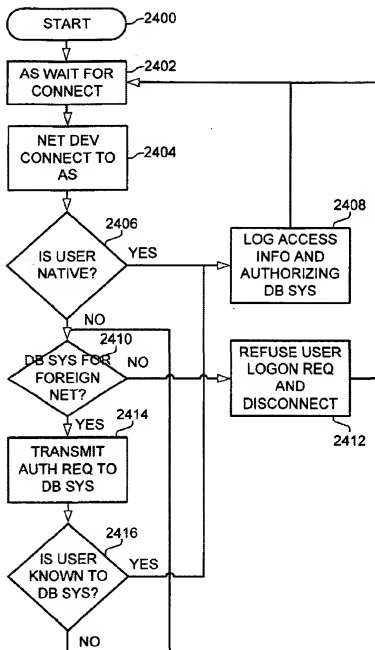


FIG. 24

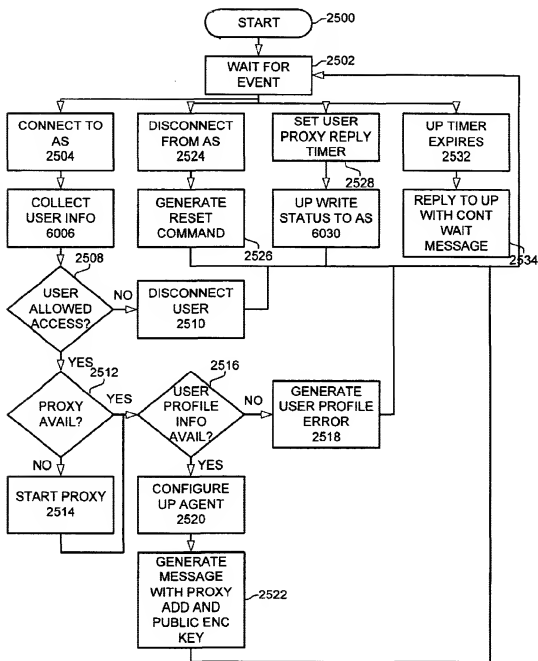


FIG. 25

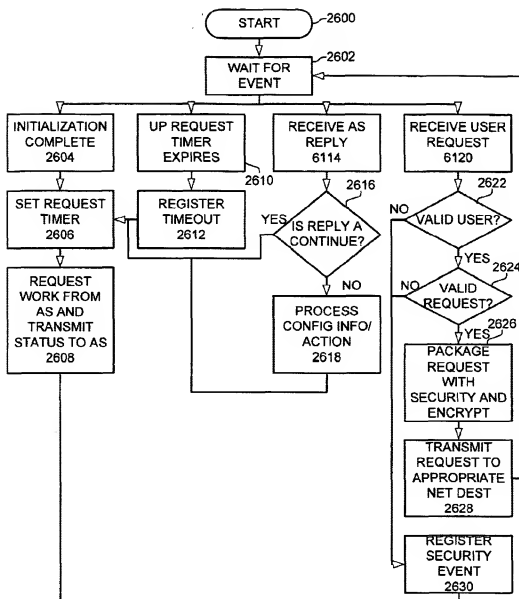


FIG. 26

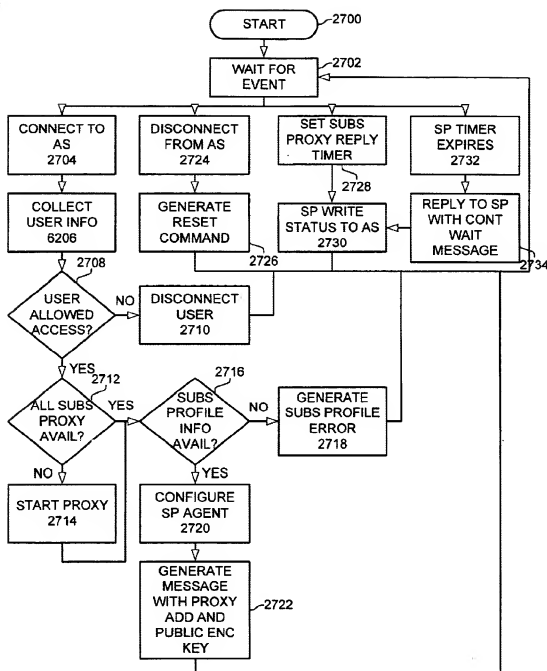


FIG. 27

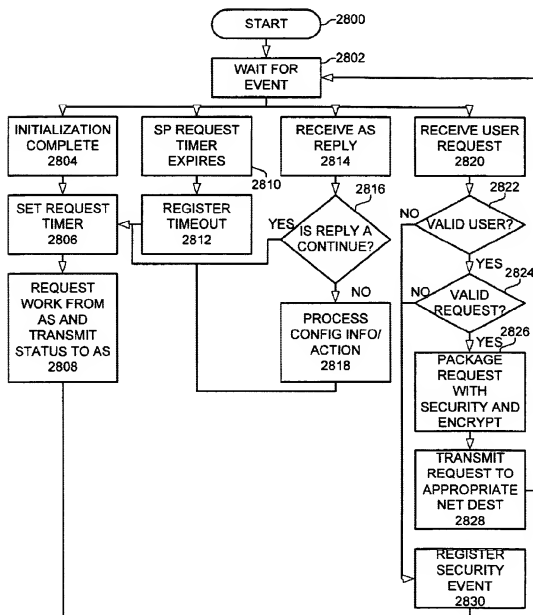


FIG. 28

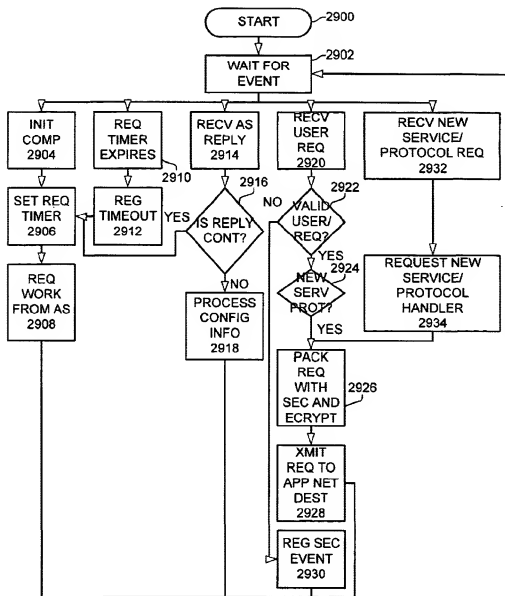


FIG. 29

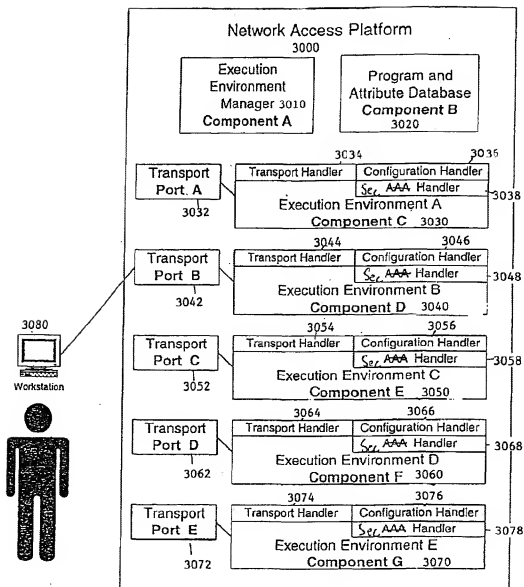


FIGURE 30

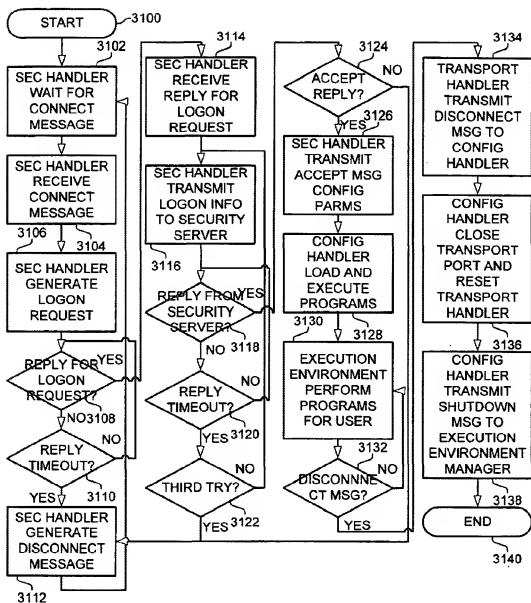


FIG. 31

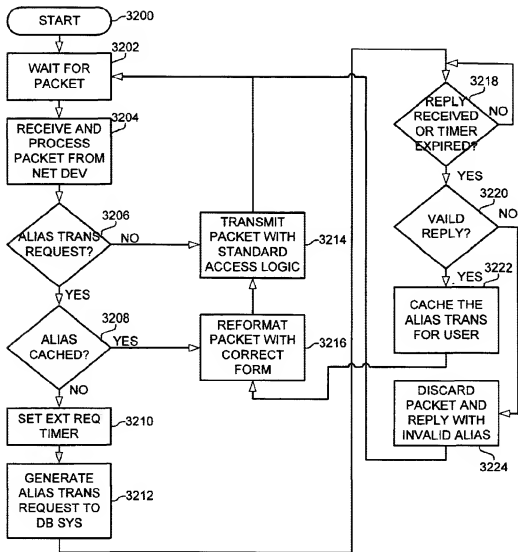


FIG. 32

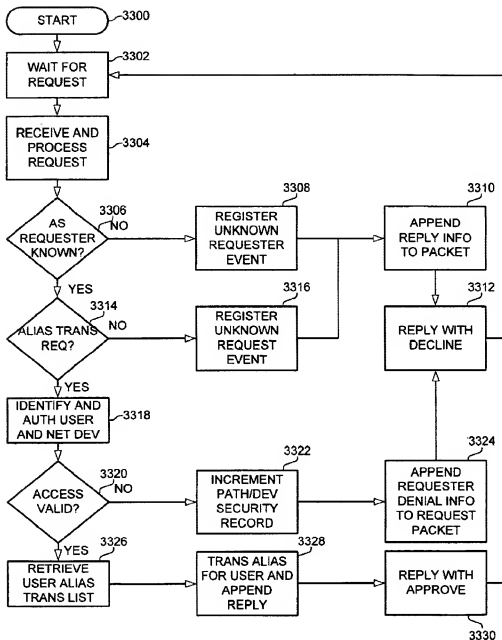


FIG. 33

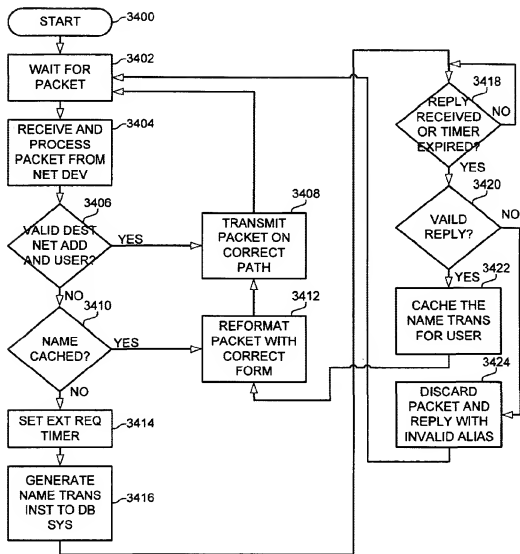


FIG. 34

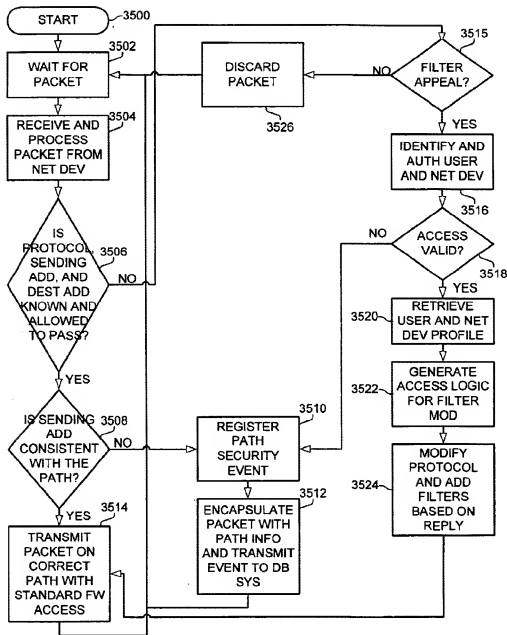


FIG. 35

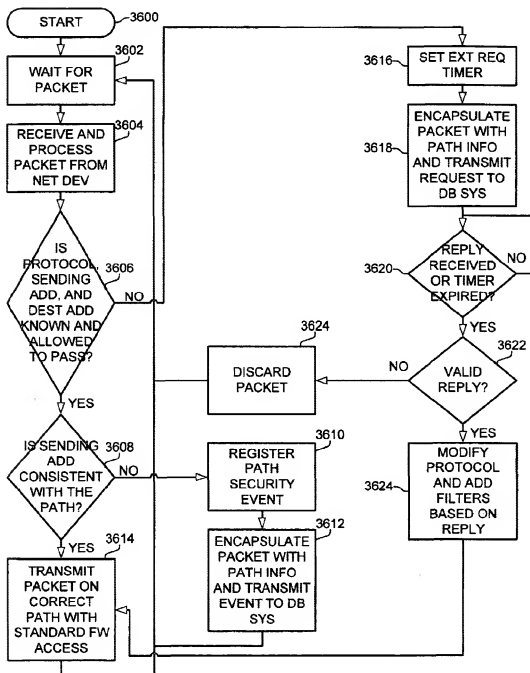


FIG. 36

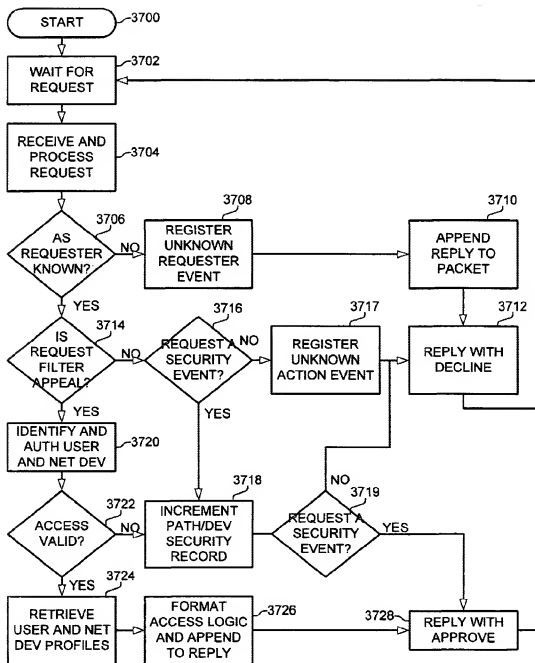


FIG. 37

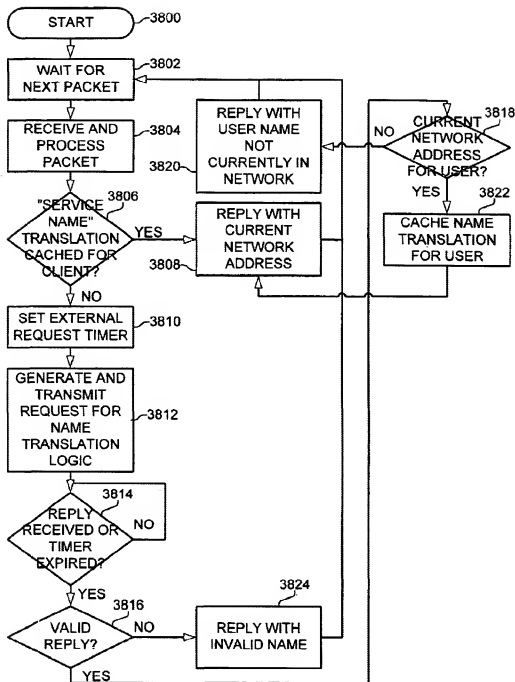


FIG. 38

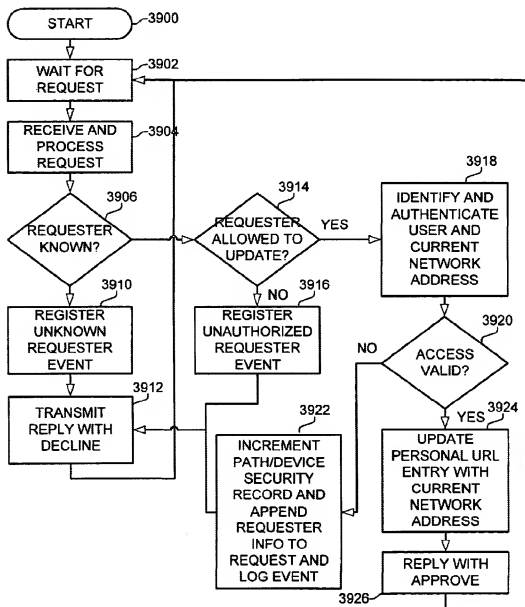


FIG. 39

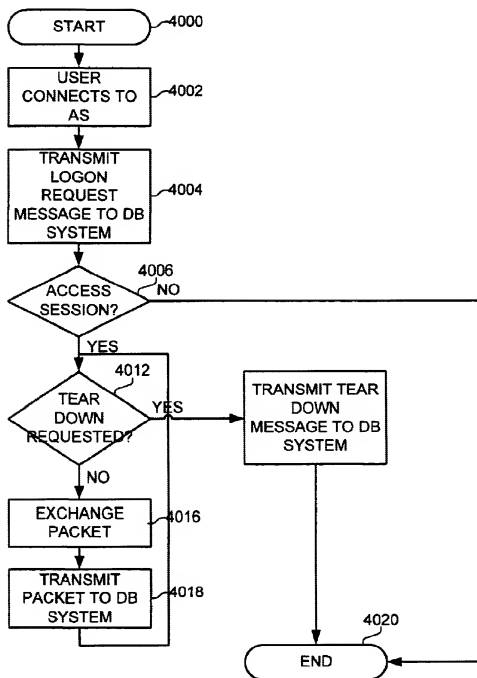


FIG. 40

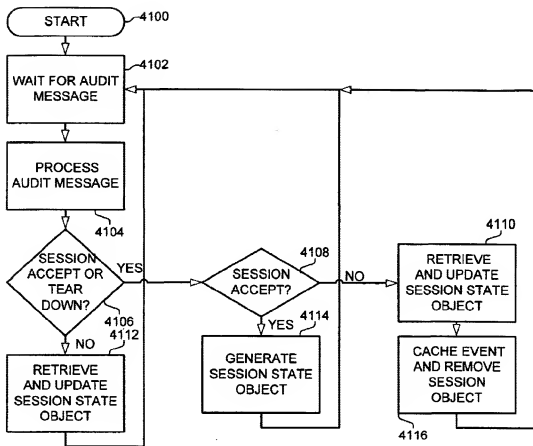


FIG. 41

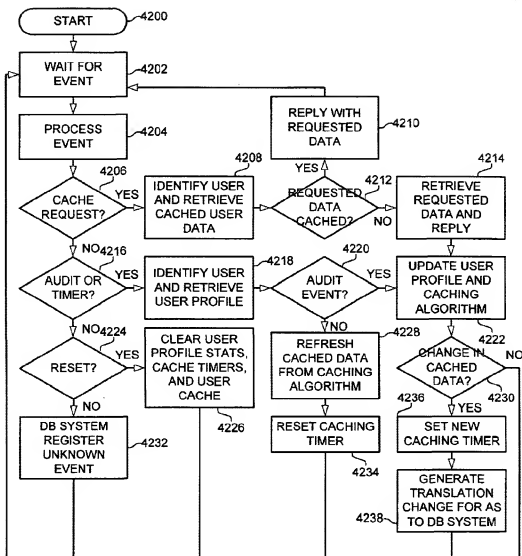


FIG. 42

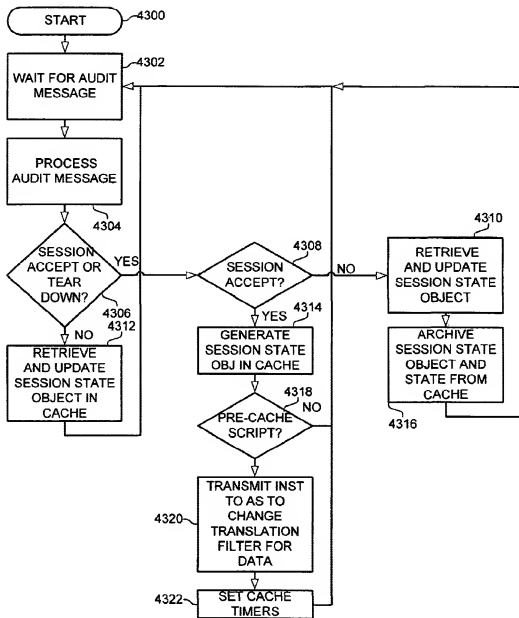


FIG. 43

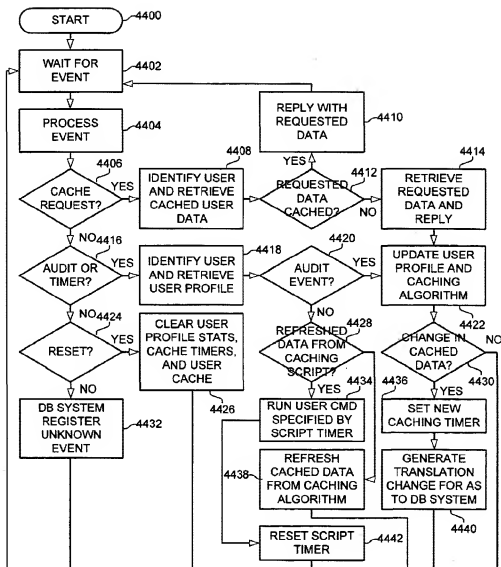


FIG. 44

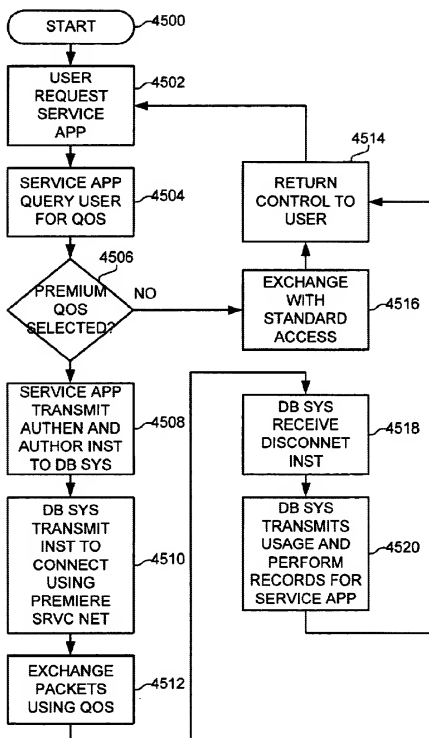


FIG. 45

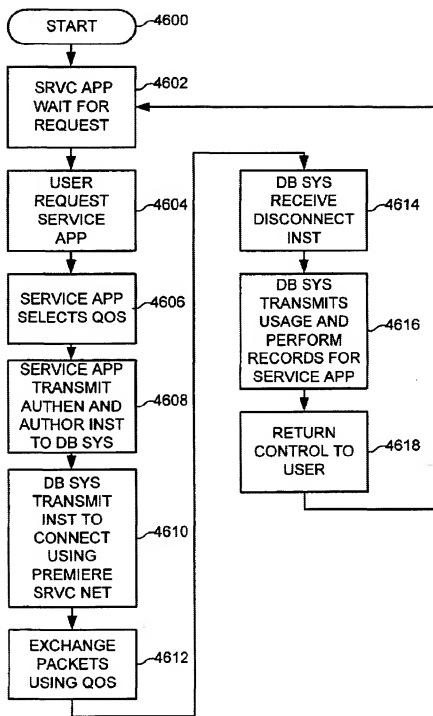


FIG. 46

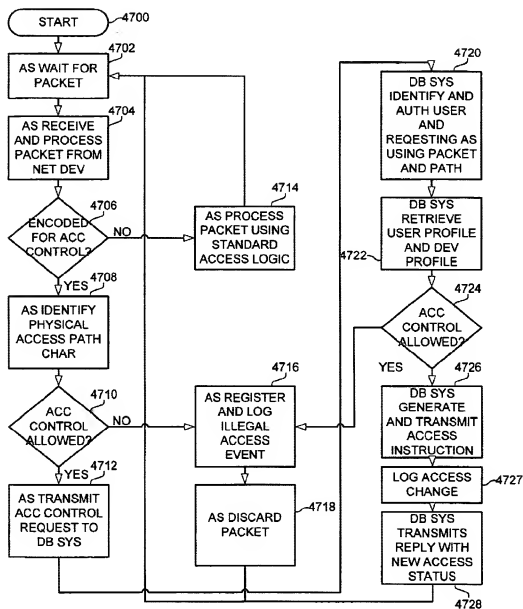


FIG. 47

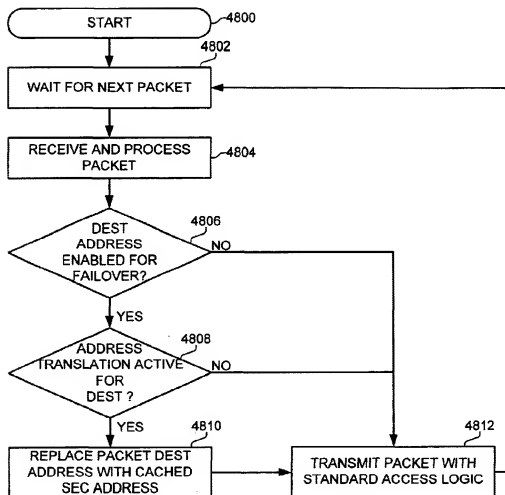


FIG. 48

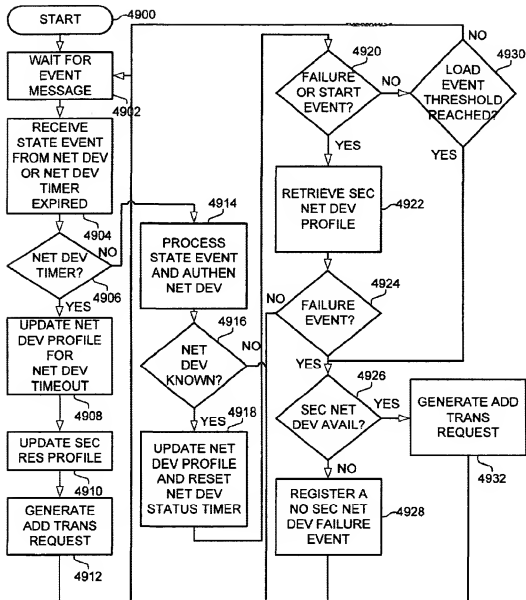


FIG. 49

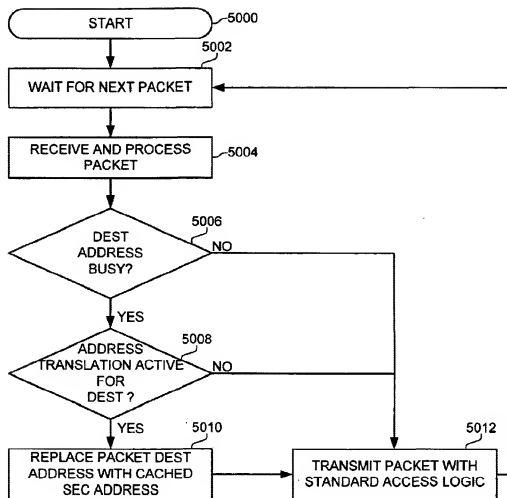


FIG. 50

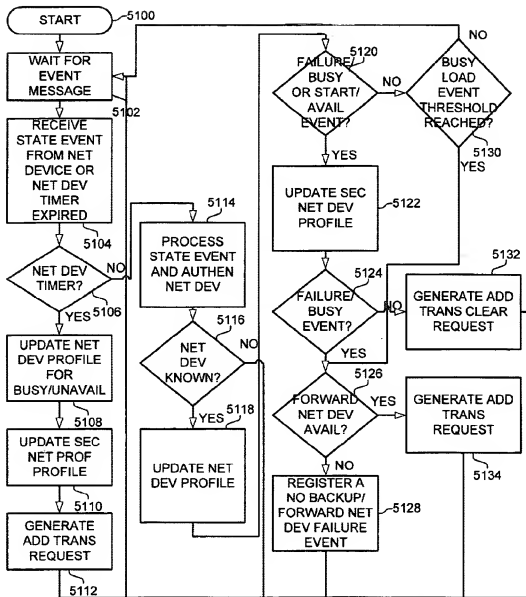


FIG. 51

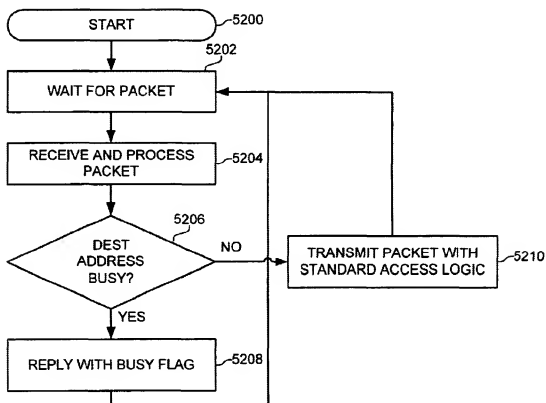


FIG. 52

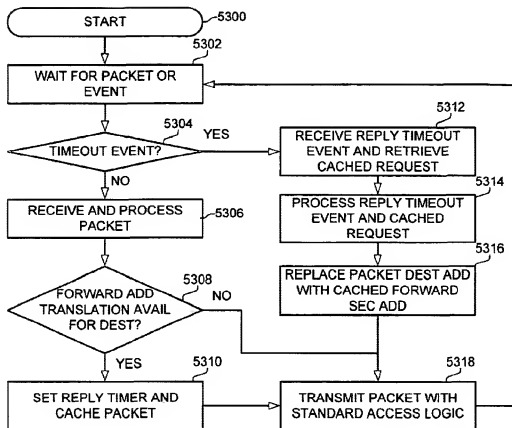


FIG. 53

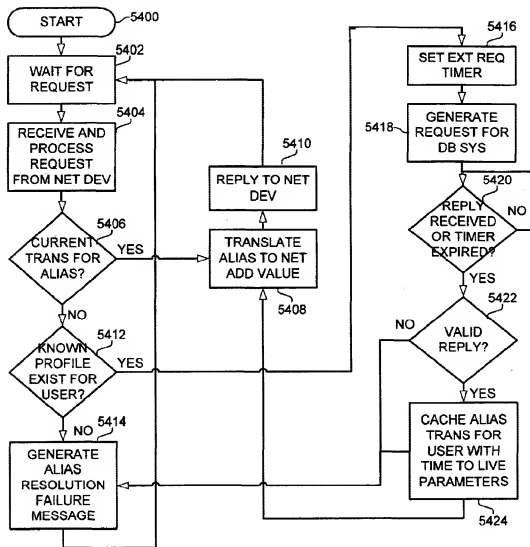


FIG. 54

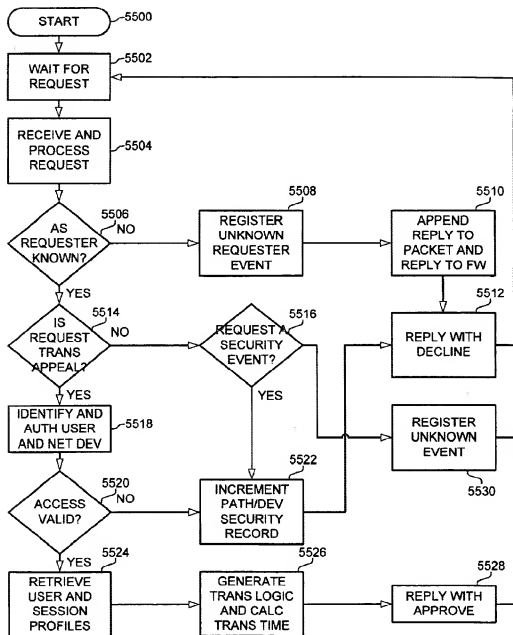


FIG. 55

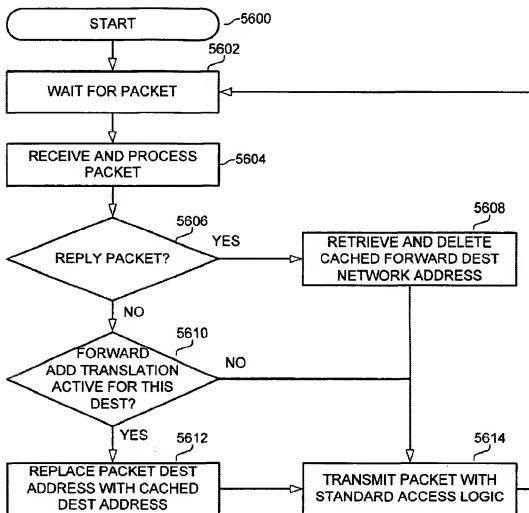


FIG. 56

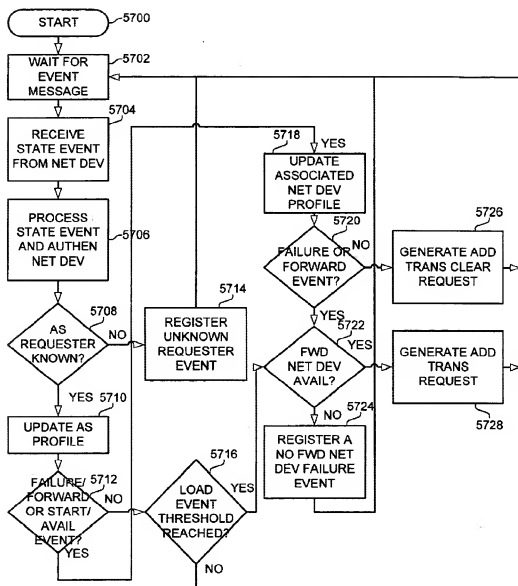


FIG. 57

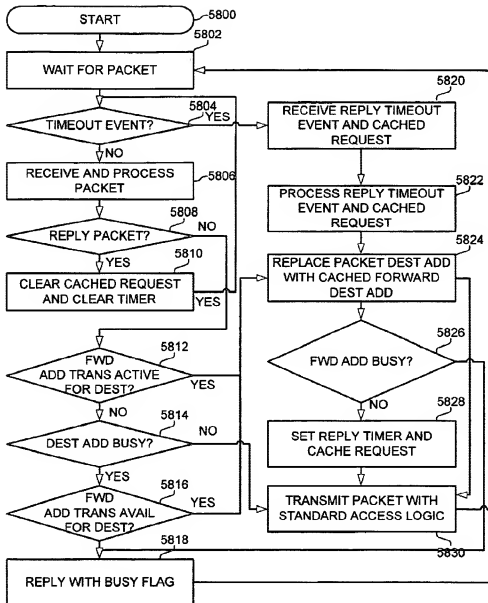


FIG. 58

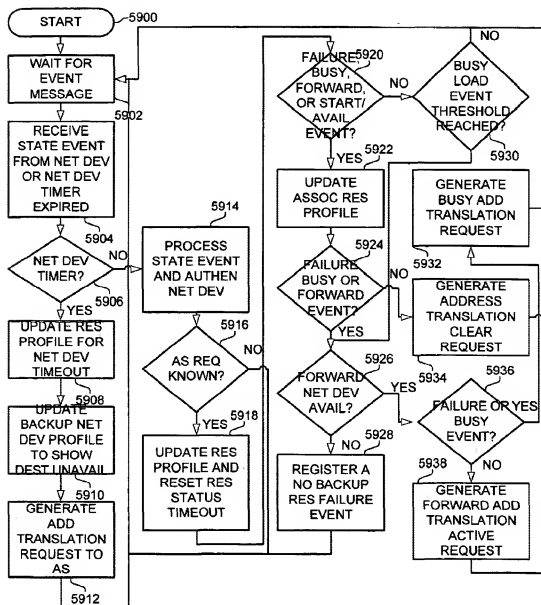
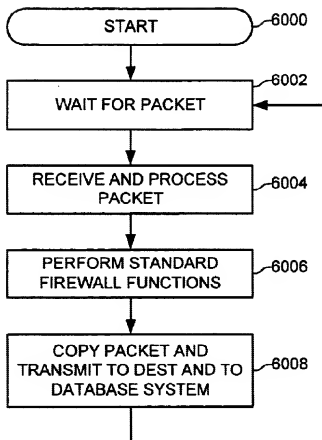


FIG. 59

**FIG. 60**

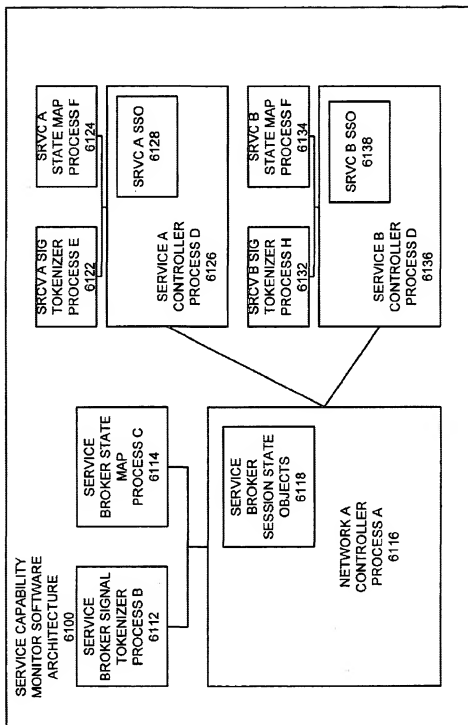
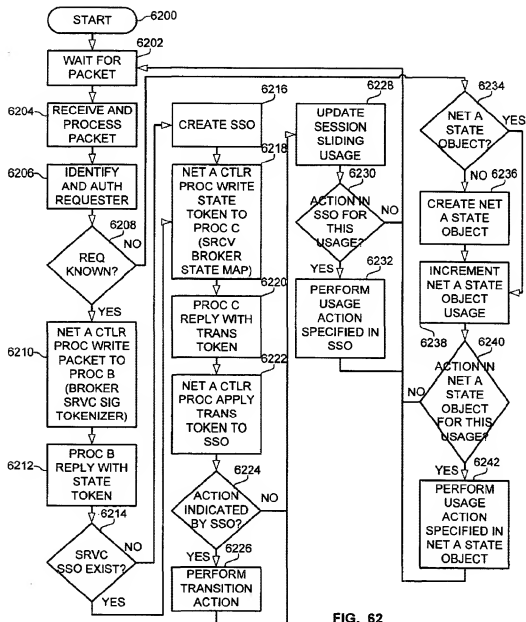


FIG. 61



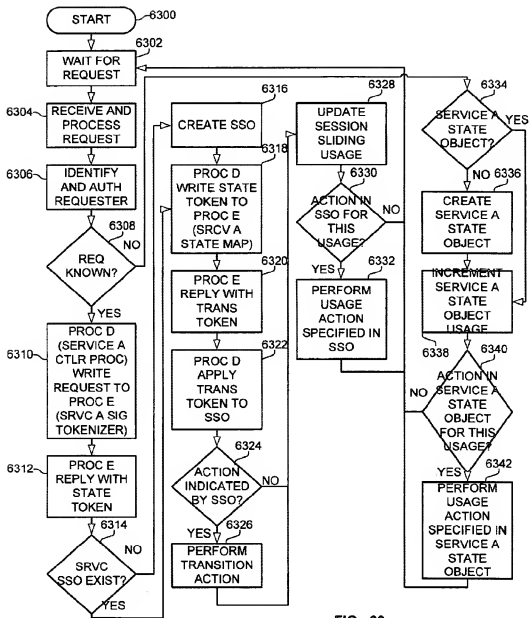


FIG. 63

ACCESS NETWORK AUTHORIZATION

RELATED APPLICATIONS

This application is a continuation of prior application Ser. No. 09/556,276, entitled "Access Communication System", filed Apr. 24, 2000, currently pending, and incorporated by reference into this application.

FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

Not applicable

MICROFICHE APPENDIX

Not applicable

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention is related to the field of communication networks, and in particular, to an access communication system that provides access to multiple service provider systems. More particularly, this invention relates to a system that authorizes access to use the access communication system.

2. Description of the Prior Art

Communication networks have seen dramatic development over the past several years so that today, there are multiple diverse communication networks providing services. The current technical challenge is to develop interfaces between the networks to provide seamless service across multiple networks. Unfortunately, today's interfaces lack the ability to offer the user with easy access to services from multiple systems. These interfaces do not customize operations for the user.

FIG. 1 illustrates the conventional public telephone network. User telephones and computers are connected to local switches. The local switches are coupled to a local database. The user places calls through the local switch. The local switch processes the called number to provide an end-point connection or access to other networks. The local switch may connect the call to another telephone in the local calling area. In a Local Number Portability (LNP) situation, the local switch exchanges information with the local database to obtain the appropriate routing number for a ported call. The local switch may also connect the call to an Internet Service Provider. If a Digital Subscriber Line (DSL) is used, DSL equipment may be used to bypass the local switch. The local switch may also connect the call to a long distance switch. To provide access to the long distance switch, the local switch first exchanges information with the local database. The local database identifies the long distance network for either the user or the dialed number.

The long distance switch processes the called number to route the call to another system or network. Prior to routing, the long distance switch validates the call by checking the caller's number. The long distance switch may also exchange information with the long distance database to provide special call-handling. One example is a calling card call, where the long distance database validates an account number for billing the call. Another example is a toll-free call, where the long distance database processes external information customized by the called party to route the call. Toll-free routing information includes items such as time and date, caller location, and call center status. Many long distance calls are simply routed through the long distance

network to another local network for call completion. Other calls are routed from the long distance network to a call center. Call centers offer a concentration of call-handling capabilities for operations, such as order entry, customer service, and promotions. Call centers include automatic call distribution equipment to route calls to the appropriate destination within the call center.

Both local and long distance networks exchange calls with mobile switches. The mobile switches are connected to base stations that communicate over the air with wireless telephones. When a mobile user places a call, the mobile switches exchange information with a mobile database to validate the mobile caller. The call is then routed to another mobile caller, the telephone network, or to an ISP. When a mobile caller moves around, their wireless phone logs-in with the physically proximate mobile network, and that mobile network updates the mobile user's location in the mobile databases. When a call is placed to that mobile user, their home mobile switch obtains a routing number from the mobile databases to route the call to the mobile network currently in communication with the mobile user.

FIG. 2 illustrates a conventional data network that transfers packets of user data to a destination based on address information carried in the packets. Users are connected to Local Area Networks (LANs) that are connected to Wide Area Networks (WANs). A common LAN is an Ethernet system. A common WAN is an intranet. WANs are interconnected by data networks, such as IP, T1, frame relay, or Asynchronous Transfer Mode (ATM). WANs are connected to the Internet through ISPs. WANs are connected to the public telephone network through telephony gateways. A common telephony gateway is a Private Branch Exchange (PBX).

FIG. 3 illustrates a conventional ISP. The public telephone network is coupled to a telephony interface that converts between telephony analog and digital protocols and the Internet Protocol (IP). Some telephony interfaces also handle DSL traffic that may already use IP. The telephony interface transfers IP traffic through an access server and firewall to a router. Some ISPs combine the firewall and the access server into one system. Also, the position of the firewall may vary, and traffic shapers may be present. The router exchanges IP traffic with the Internet.

In operation, the user calls the ISP over the telephone network and logs-in at the access server. The access server collects and forwards the user name and password to the ISP database. The ISP database validates the user name and password and returns an IP address to the access server. The IP address is for the user's terminal connection. Using the IP address, the user may communicate through the firewall to the router for transmissions to an IP address. The user now has Internet access through the router and exchanges packets with various Internet servers.

IP addresses are referred to as network addresses and include a network ID and a host ID. Network IDs are unique across the Internet and host IDs are unique within a given network. IP addresses are lengthy numerical codes, so to simplify things for the user, service addresses are available that are easier to remember. The service addresses are often the name of the business followed by ".com". Domain Name Service (DNS) is hosted by servers on the Internet and translate between service addresses and network addresses. The browser in the user computer accesses the DNS to obtain the desired network address.

FIG. 4 illustrates conventional network access. A current proposal for communication network access is provided by

3

the Telecommunication Information Network Architecture Consortium (TINA-C). TINA-C proposes the use of agents in the user domain and the service provider domain. The service provider domain could be a telephone network, data network, or ISP. The agents negotiate access service rights. Once the service is negotiated, the user receives the service from the service provider network during a service session. Unfortunately, the access session occurs between the user domain and a particular service provider domain. At present, the service provider domain provides limited access capability beyond simply handing off communications to another network based on a called number or network address. As a result, the ability to customize services for a particular user across multiple service providers is inadequate.

SUMMARY OF THE INVENTION

The inventions solve the above problems by providing access between a user system and a plurality of communication networks. The plurality of communication networks provide services to a user in the user system. An access communication system includes a database system and an access server that is connected to the user system and the plurality of communication networks.

In one aspect of the inventions for user access profile inheritance, the database system receives an update request from the access server to update a user access profile through inheritance. The database system then processes the update request to inherit user profile information from a user profile data structure. The database system updates the user access profile with the user profile information.

In another aspect of the inventions for network shells, the access server receives an alias selection from a user for a network shell that includes alias selections associated with actions. The access server then processes the alias selection to execute an action associated with the alias selection.

In another aspect of the inventions for service based directory, the access server transmits a list of services to a user system. The access server then receives a selection from the list of services. The access server processes the selection to generate an instruction to provide the service related to the selection.

In another aspect of the inventions for user access profile mobility, the database system receives user information. The database system then processes the user information to determine if a user access profile is local within a local database system. The database system generates and transmits a request to retrieve a user access profile from a second database system external to the local database system in response to the determination that the user access profile is not local.

In another aspect of the inventions for service, user, and device sessions, the access communication system establishes a connection between a network device and the access server. The access communication system then generates a device session including a device session ID based on the network device. The access communication system generates and transmits a login query for the network device. The access communication system receives and processes a login reply from the network device to generate a user session including a user session ID based on the user. The access communication system receives and processes a request for the service to generate a service session including a service session ID based on the service. The service may generate and transmit a login query for the user. The access communication system links the device session, user session, and the service session using the device session ID, the user session ID, and the service session ID.

4

In another aspect of the inventions for service capability firewall, the access server receives information including a named function request for a service provider. The access server processes the information to check if the named function request is valid for the service provider and the service. If valid, the access server determines if a private destination address exists for the named function request. The access server replaces the named function request with the private destination address in response to the determination that the private destination address exists for the named function request. The access server then transmits the information with the private destination address to the service provider.

In another aspect of the inventions for prepaid access and bank card access, the database system receives information identifying a billing code for a user. The database system then processes the billing code to determine if the user is allowed to use the access system. The database system provides access to the access system in response to the determination that the user is allowed to use the access system.

In another aspect of the inventions for global authentication and access card, the database system receives a user login. The database system then processes the user login to determine if the user is allowed access to the access communication system based on a local database system. The database system then provides access to the access communication system to the user in response to the determination that the user is allowed access based on the local database system. The database system then generates an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system. The database system receives and processes an authorization response indicating whether the user is allowed to use the access system from the second database system. The database system then provides access to the access communication system to the user in response to the authorization response that allows the user to use the access communication system.

In another aspect of the inventions for user based proxy and subscriber based proxy, the database system includes a user proxy. The user proxy receives a request for the service from the user system. The user proxy then transmits the request for the service to a service provider. The user proxy exchanges user information between the user system and the service provider.

In another aspect of the inventions for dynamic proxy, the database system includes a proxy. The proxy receives a service/protocol request for a new service or protocol. The proxy processes the service/protocol request to generate a handler request to obtain a handler for the new service or protocol. The proxy then receives and executes the handler for the new service or protocol.

In another aspect of the inventions for access execution environment, the database system receives and processes a login reply into an access execution environment for a user. The database system retrieves programs for the user into the access execution environment. The database system executes the programs for the user in the access execution environment.

In another aspect of the inventions for domain name scoping and inband domain name service lookup, the access server receives information including an alias from the user system. The access server determines if the alias exists in a cache including aliases and alias translations for the user.

5

The access server changes the information based on the cached alias translation.

In another aspect of the inventions for inline access service triggering, the access server receives information. The access server then processes the information to determine if the information is allowed to pass. The access server changes access logic based on the information in response to the determination that the information is not allowed to pass. The access server changes the filters of the access server based on the information in response to the determination that the information is not allowed to pass.

In another aspect of the inventions for access service triggering, the access server receives information. The access server processes the information to determine if the information is allowed to pass. The access server then generates a request from a database system in response to the determination that the information is not allowed to pass. The access server receives a reply including access logic from the database system. The access server changes filters of the access server based on the access logic.

In another aspect of the inventions for personal URL, the database system receives information including a user alias. The database system processes the information to determine if a user alias translation including a current network address for the user alias exists. The database system then modifies the information with the current network address using the user alias translation.

In another aspect of the inventions for predictive caching, the access server receives a request for data. The access server then determines if the data exists in a user cache wherein the user cache contains cached data based on the user's predictive patterns. The access server retrieves the data from the user cache in response to the determination that the data exists in the user cache. The access server transmits the request for data for the service provider in response to the determination that the data does not exist in the user cache.

In another aspect of the inventions for user controlled caching, the access server receives a request for data. The access server determines if the data exists in a user cache wherein the user cache contains cached data based on a user's script of commands. The access server retrieves the data from the user cache in response to the determination that the data exists in the user cache. The access server then transmits the request for data for the service provider in response to the determination that the data does not exist in the user cache.

In another aspect of the inventions for service usage audit, the access server receives an audit message into a database system. The access server processes the audit message to store the audit message in the database system.

In another aspect of the inventions for switching access by a user, switching access by a service provider, and dynamic access control, the database system receives a request. The database system processes the request to determine if the switching of the access is allowed. The database system then generates an instruction to switch access in response to the determination that the switching is allowed.

In another aspect of the inventions for network failover, network busy forwarding, time-out, busy flag, forwarding, and network endpoint availability management, the access server receives information for a destination network device. The access server determines if the destination network device is available. The access server performs an action in response to the determination that the destination network device is unavailable.

6

In another aspect of the inventions for scheduled alias translation, the access server receives information including an alias. The access server processes the information to determine whether an alias translation exists based on an alias translation schedule. The access server then modifies the information based on the alias translation in response to the determination the alias translation exists.

In another aspect of the inventions for service capability monitor, the database system receives information from an access server during a service session. The database system determines a current state of the service session based on the information. The database system determines a state transition based on the current state and a map of state transitions of the service. The database system determines whether the state transition is valid for the service session.

BRIEF DESCRIPTION OF THE DRAWINGS

The same reference number represents the same element on all drawings.

FIG. 1 illustrates a public telephone network in the prior art.

FIG. 2 illustrates a data network in the prior art.

FIG. 3 illustrates an Internet Service Provider in the prior art.

FIG. 4 illustrates conventional network access.

FIG. 5 illustrates a network architecture in an example of the invention.

FIG. 6 illustrates an access network in an example of the invention.

FIG. 7 illustrates a table for a user access profile in an example of the invention.

FIG. 8 illustrates a flowchart for an access server for inheriting a user access profile in an example of the invention.

FIG. 9 illustrates a flowchart for a database system for inheriting a user access profile in an example of the invention.

FIG. 10 illustrates a flowchart of an access server for executing a network shell in an example of the invention.

FIG. 11 illustrates a flowchart of a database system for updating a network shell in an example of the invention.

FIG. 12 illustrates a flowchart for the services based directory in an example of the invention.

FIG. 13 illustrates a flowchart of user access profile mobility in an example of the invention.

FIG. 14 illustrates a logical view of device, user, and service sessions in an example of the invention.

FIG. 15 illustrates a flowchart for a user based session in an example of the invention.

FIG. 16 illustrates a flowchart for a service based session in an example of the invention.

FIG. 17 illustrates a flowchart for a firewall/router for service capability firewall in an example of the invention.

FIG. 18 illustrates a flowchart for a database system for service capability firewall in an example of the invention.

FIG. 19 illustrates a flowchart for prepaid access in an example of the invention.

FIG. 20 illustrates a flowchart for bank card access for a connection in an example of the invention.

FIG. 21 illustrates a flowchart for network access cards for a disconnection in an example of the invention.

FIG. 22 illustrates a flowchart for network access cards for a connection in an example of the invention.

7

FIG. 23 illustrates a flow chart for network access cards for a disconnection in an example of the invention.

FIG. 24 illustrates a flowchart for global access in an example of the invention.

FIG. 25 illustrates a flowchart for an access server for user based proxies in an example of the invention.

FIG. 26 illustrates a flowchart for a user proxy for user based proxies in an example of the invention.

FIG. 27 illustrates a flowchart for an access server for subscriber based proxies in an example of the invention.

FIG. 28 illustrates a flowchart for a subscriber proxy for subscriber based proxies in an example of the invention.

FIG. 29 illustrates a flowchart for a dynamic proxy for dynamic proxies in an example of the invention.

FIG. 30 illustrates a block diagram of the access execution environment in an example of the invention.

FIG. 31 illustrates a flow chart for the access execution environment in an example of the invention.

FIG. 32 illustrates a flowchart for an access server for domain name scoping in an example of the invention.

FIG. 33 illustrates a flowchart for a database system for domain name scoping in an example of the invention.

FIG. 34 illustrates a flowchart for an access server for an inband domain name service lookup in an example of the invention.

FIG. 35 illustrates a flowchart for inline access service triggering in an example of the invention.

FIG. 36 illustrates a flowchart for an access server for access service triggering in an example of the invention.

FIG. 37 illustrates a flowchart for a database system for access service triggering in an example of the invention.

FIG. 38 illustrates a flow chart for the personal URL lookup in an example of the invention.

FIG. 39 illustrates a flow chart for the personal URL update in an example of the invention.

FIG. 40 illustrates a flowchart for an access server for auditing in an example of the invention.

FIG. 41 illustrates a flowchart for a database system for auditing in user predictive caching in an example of the invention.

FIG. 42 illustrates a flowchart for a database system for caching in user predictive caching in an example of the invention.

FIG. 43 illustrates a flowchart for a database system for auditing in user controlled caching in an example of the invention.

FIG. 44 illustrates a flowchart for a database system for caching in user controlled caching in an example of the invention.

FIG. 45 illustrates a flowchart for switching access by a user in an example of the invention.

FIG. 46 illustrates a flowchart for switching access by a service provider in an example of the invention.

FIG. 47 illustrates a flowchart for dynamic switching access in an example of the invention.

FIG. 48 illustrates a flowchart for an access server for network address failover in an example of the invention.

FIG. 49 illustrates a flowchart for a database system for network address failover in an example of the invention.

FIG. 50 illustrates a flowchart for an access server for network busy forwarding in an example of the invention.

FIG. 51 illustrates a flowchart for a database system for network busy forwarding in an example of the invention.

8

FIG. 52 illustrates a flowchart for an access server for a busy flag when the destination network device is busy in an example of the invention.

FIG. 53 illustrates a flowchart for an access server for forwarding, if the destination network device timeouts in an example of the invention.

FIG. 54 illustrates a flowchart for an access server for schedule alias resolution in an example of the invention.

FIG. 55 illustrates a flowchart for a database system for scheduled alias resolution in an example of the invention.

FIG. 56 illustrates a flowchart for an access server for destination controlled forwarding in an example of the invention.

FIG. 57 illustrates a flowchart for a database system for destination controlled forwarding in an example of the invention.

FIG. 58 illustrates a flowchart for an access server for network endpoint availability management in an example of the invention.

FIG. 59 illustrates a flowchart for a database system for network endpoint availability management in an example of the invention.

FIG. 60 illustrates a flowchart for a firewall/router for service capability monitor in an example of the invention.

FIG. 61 illustrates a service capability monitor software architecture for a service capability monitor in an example of the invention.

FIG. 62 illustrates a flowchart for the network logic for service capability monitor in an example of the invention.

FIG. 63 illustrates a flowchart for the service logic for service capability monitor in an example of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Communication Network Architecture—FIGS. 5 and 6

The following description and associated figures discuss specific examples intended to teach the present invention to those skilled in the art. Those skilled in the art will appreciate numerous variations from these examples that do not depart from the scope of the invention. Those skilled in the art will also appreciate that various features described below could be combined in various ways to form multiple variations of the invention.

FIG. 5 illustrates a network architecture 500 in an example of the invention. The network architecture 500 comprises a service network 530, a service network 540, and an access network 520. The access network 520 comprises a database system 522, an access server 524, a firewall/router 526, a firewall/router 556, and an access server 554. A user network 510 includes a network device 512. The user network 510 is connected to the access server 524. The access server 524 is connected to the firewall/router 526 and the database system 522. The firewall/router 526 is connected to the service network 530, the service network 540, and the database system 522. The firewall/router 556 is connected to the service network 530, the service network 540, the database system 522, and the access server 554. The access server 554 is connected to the database system 522 and the user network 560. The user network 560 comprises a network device 562 and a network device 564.

The access network 520 provides an interface between the user network 510 and 560 and the service networks 530 and

540. The interface function provides user access profiles, security, switching, and caching. The user network 510 and 560 could be a residential or business communication system that includes network devices. A network device 512, 562, and 564 could be any device configured to exchange data or information with the access network 520. Some examples of network devices are wireless and wireline telephones, computers, modems, servers, and/or data terminals, along with associated interconnections and network interfaces. For illustrative purposes in the examples below, a user interacts with the network device 512 to access services provided by the network architecture 500 and the user network 560. The user network 510 could also be a communication destination for other users. In these cases, the access network 520 provides destination performance monitoring and control. The example of a user interacting with the network device 562 and 564 to access services provided by the network architecture 500 and the user network 510 is not discussed below for the sake of clarity.

The access server 524 and the database system 522 could be adapted from components used in current ISPs, and the access network 520 could be integrated into these ISP components. The database system 522 houses user access profiles for the users in the user networks 510 and 560 with external access rights. The user access profiles provide a data clearinghouse for user-related information for security, service options, current state, and customized macros. The service networks 530 and 540 could be voice or data systems, such as the public telephone network, Internet, public data networks, and private data networks.

When a user requests access to services, the access network 520 processes the user access profile for the user. The access network 520 performs security measures to validate the user. The access network 520 then binds the user to a terminal and to a service. This user-terminal-service binding correlates the user's capabilities in the user access profile with the capabilities of the user's current terminal and the current capabilities of the service provider. The access network 520 activates a network shell for the user and the user's terminal. The network shell is an interface that is customized for the user and the terminal. The user invokes services using the network shell.

The user access profile may contain several macros that can be as simple as address translations or as complex as lengthy computer programs. The user's network shell provides access to the user's macros. The user access profile also stores caching instructions. The caching instructions control the collection and storage of information within the access network 520 for immediate access by the user.

FIG. 6 depicts an access network in an example of the invention. The access network 520 comprises the database system 522, the access server 524, the firewall/router 526, the firewall/router 556, and the access server 554. The database system 522 comprises a local database system 570, a Lightweight Directory Access Protocol (LDAP) interface system 571, a central database system 580, a local database system 590, and an LDAP interface system 591. The local database system comprises a user profile system 572, an audit database system 573, a cache database system 574, a host system 575, a security server 576, a user authorization system 577, an alias translation system 578, and a personal DNS system 579. The central database system 580 comprises a user authorization system 581, a financial interface 582, and a cross connect system 583. The local database system 590 comprises a user profile system 592, a user authorization system 593, and an availability system 594.

The access server 524 and the firewall/router 526 are connected to the local database system 570. The local

database system 570 is connected to the central database system 580 and the LDAP interface system 571. The central database system 580 is connected to the local database system 590. The local database system 590 is connected to the firewall/router 556, the access server 554, and the LDAP interface system 591.

An access provider is an entity that provides access to users who use communications services. A typical access provider comprises an access server, a firewall/router, and a local database system. The systems within the local database system 570 could be included within the local database system 590. Also, the systems within the local database system 590 could be included within the local database system 570. As discussed above, the user is accessing the network architecture through the user network 510. The duplication of systems in the local database system 570 and the local database system 590 is excluded for the purposes of clarity. A service provider is an entity that provides communication services to users who are accessing the service through an access provider.

Network User Access Profile—FIGS. 5, 6, 7, 8, 9, 10, 11, and 12

User Access Profile Inheritance

The user access profile is stored in the access network 520 and controls user access to services. The user access profile is any information or data associated with controlling user access to a service such as role identification, authorization, billing information and access preferences.

FIG. 7 depicts table for a user access profile in an example of the invention. A user access profile includes access information for the user, billing information, and preferences for access. Access information is any information or data related to providing the user access to the network architecture 500. Some examples of access information are user ID, password, name, account number, user alias, current network address, switching allowed flag, and other security information. Access information also include a list of services that the user has subscribed to or is allowed access to. In one embodiment, access information includes a cache of information that the user has accessed previously. Some examples of billing information are address and billing code including bank card numbers or prepaid account codes. Preferences for access allow the user to save choices or preferences to customize their access to the network architecture 400. Some examples of preferences for access are file formats and Quality of Service values. The user access profile may also include usage information such as time of day access, day of week, usages per day, usages per week, and usages per month.

Users typically set up their user access profiles when signing up for the access to the network architecture 500. In one embodiment, users create the user access profile through an inheritance process. Through the inheritance process, the user selects user profile information from other user access profiles in the network architecture 500. The user may inherit user profile information from user profile data structures such as templates, other user access profiles, the user's group or class, or other networks that the user is set up in. A user profile data structure is any user profile information to be retrieved when inheriting a user access profile.

In a prior solution, a command interpreter in a single computer operating system shell environment allows a user to customize the interpretation of command strings. The user may inherit portions of other user's shell environment by adding the other user's shell attributes. User access profiles are typically stored in the access provider's database. FIGS.

5 and FIG. 8 disclose one embodiment for inheriting user access profiles in an example of the invention. The user access profile is created through an inheritance process where the user is able to select capabilities, macros, functions, methods, and data to inherit from other profiles. In this embodiment, users inherit user access profiles from classes, groups, or provider recommendations. Features of user access profiles such as alias translation could then be implemented with new users rapidly. The inheritance of user access profiles also simplifies the configuration of users. In one example, a user initiates the user access profile inheritance by clicking a button on a website. Also, when a user requests a new service, the service provider automatically inherits the user access profile for the user so the user is able to use the requested service.

FIG. 8 depicts a flowchart for the access server 524 for inheriting a user access profile in an example of the invention. FIG. 8 begins in step 800. In step 802, the access server 524 waits for the next packet. The access server 524 then receives and processes the packet from the network device 562 in step 804. The access server 524 then checks if the destination is the user access profile in step 806. If the destination is not the user access profile, the access server 524 transmits the packet on the correct path in step 808 before returning to step 802. If the destination is the user access profile, the access server 524 then checks if the network device 562 is allowed access to the user access profile in step 1110. If the network device 512 is not allowed access to the user access profile, the access server 524 discards the packet and registers a network request security event in step 812 before returning to step 802.

If the network device 512 is allowed access to the user access profile, the access server 524 then checks if the user is allowed to update their user access profile in step 814. If the user is not allowed updates to their user access profile, the access server 524 generates and transmits a profile update not allowed message to the network device 512 in step 816 before returning to step 802. If the user is allowed updates to their user access profile, the access server 524 generates and transmits a user access profile update permission message asking if the user wishes to update the user access profile to the network device 512 in step 818. The access server 524 then checks if the user approved the user access profile update in step 820. If the user does not approve, the access server 524 generates and transmits a user access profile aborted message to the network device 512 in step 822 before returning to step 802.

If the user approves the user access profile update, the access server 524 sets an external request timer in step 824. The access server 524 then generates and transmits a user access profile update request to the database system 522 in step 826. A user access profile update request could be any signaling, message, or indication to update the user access profile. The access server 524 then checks if a reply was received or the external request timer expired in step 828. If the reply was not received and the external request timer did not expire, the access server 524 returns to step 828. If the reply was received or the external request timer did expire, the access server 524 checks if the reply was valid in step 830. If the reply was valid, the access server 524 generates an update complete message in step 832 before returning to step 802. If the reply was not valid, the access server 524 discards the packet and replies to the network device 512 that the user access profile update failed in step 834 before returning to step 802.

FIG. 9 depicts a flow chart for the database system 522 for inheriting a user access profile in an example of the inven-

tion. FIG. 9 begins in step 900. In step 902, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 904. In step 906, the database system 522 then checks if the access server requester is known. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 912. The database system 522 appends the reply to the packet and transmits the reply to the access server 524 in step 910. The database system 522 then replies with a decline message to the access server 524 in step 912 before returning to step 902.

If the requester is known, the database system 522 then checks whether the request is a user access profile update request in step 914. If the request is not a user access profile update request, the database system 522 checks if the request is a security event in step 916. If the request is not a security event, the database system 522 then registers an unknown action event in step 918 before returning to step 910. If the request is a security event, the database system 522 increments a path/device security record in step 924. The database system 522 then logs the path/device security record in step 926 before returning to step 902.

If the request is a user access profile update request, the database system 522 identifies and authenticates the user and the network device 512 in step 920. The database system 522 then checks if the access is valid in step 922. If the access is invalid, the database system 522 returns to step 924. If the access is valid, the database system 522 retrieves the user access profile information from a user profile data structure in step 928. The retrieval of user access profile information may be based on any information in the user access profile such as group or class or selections of inheritance user access profile presented to the user. The group or classes could be any logical grouping of user based on similar interests or situations such as work, family, and geographic location. The database system 522 then updates the user access profile in step 930 based on the user access profile information retrieved in step 928 and the user's selections for updating. The database system 522 then replies with an approve message to the access server 524 in step 932 before returning to step 902. In one embodiment of the invention, the database system 522 uses the user access profile system 572 to perform the operations disclosed in FIG. 9 and stores the user access profile in the user access profile system 572.

45 Network Shell

The user access profile includes data to provide a customized network shell to the user. The network shell is a user interface to the access network that is linked to programs, methods, macros, or service. Typically, configuration of the network shell is graphically presented to the user on a display. For example, the network shell appears as a list of alias selections that are associated with actions such as a program or macro for resources or services. An alias selection is any information that is associated with an action to be executed when the alias selection is selected. In another example, the alias selections are graphically presented as icons that relate to actions to be executed. The network shell overlays the standard DNS offered in IP networks, which reduces alias translation delays and required user keystrokes. In prior solutions, a command interpreter in an operating system shell environment allows a user to customize the interpretation of command strings, or a DNS translates "alias" values to addresses. FIGS. 5, 10, and 11 show one embodiment for network shells in an example of the invention.

FIG. 10 depicts a flowchart of an access server for executing a network shell in an example of the invention.

13

FIG. 10 begins in step 1000. In step 1002, the access server 524 waits for the next packet. The user selects an alias selection from the network shell graphically presented. In another embodiment, the user enters an alias value by a non-graphical means. Alternatively, the network shell is not presented to the user. The network device 512 transmits a packet including the alias selection to the access server 524. In step 1004, the access server 524 receives and processes the packet from the network device 512 and processes the packet. The access server 524 then checks if the network device 512 is allowed access to the network architecture 500. If the network device 512 is not allowed access to the network architecture 500, the access server 524 discards the packet and registers a network security event in step 1008 before returning to step 1002.

If the network device 512 is allowed access to the network architecture 500, the access server 524 checks if the user is recognized in step 1010. If the user is not recognized, the access server 524 returns to step 1008. If the user is recognized, the access server 524 retrieves the user's network shell in step 1012. In some embodiments, the user's network shell is retrieved from the user access profile in the access database 522 prior to presenting the network shell to the user. In step 1014, the access server 524 checks if the packet from the network device 512 includes an alias selection from the user's network shell. In one embodiment, specialized hardware is used to scan for aliases just as IP addresses are currently scanned for. If the packet does not include an alias selection from the user's network shell, the access server 524 proceeds to step 1018. If the packet does include the alias selection from the user's network shell, the access server 524 executes the action associated with the alias selection in step 1016 before returning to step 1002. In step 1018, the access server 524 processes the packet with the normal handling before returning to step 1002.

FIG. 11 depicts a flowchart of a database system for updating a network shell in an example of the invention. FIG. 11 begins in step 1120. In step 1122, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 1124. The database system 522 then checks if the access server requester is known in step 1126. If the access server requester is not known, the database system 522 registers an unknown requester event in step 1128. The database system 522 then appends the reply to the packet and transmits the packet to the access server 524 in step 1130. In step 1132, the database system 522 replies with a decline message to the access server 524 before returning to step 1122.

If the access server requester is known, the database system 522 checks if the request is a profile update for the network shell in step 1134. If the request is a profile update for the network shell, the database system 522 identifies and authenticates the user and the network device 512 in step 1136. The database system 522 then checks whether the access is valid in step 1138. If the access is invalid, the database system proceeds to step 1154. If the access is valid, the database system 522 retrieves the user access profile in step 1140. The database system 522 then updates the user access profile with the network shell with the user's alias selections and the associated programs, macros, functions or methods in step 1142. The database system 522 then replies with an approve message in step 1144 before returning to step 1122.

If the request is not a profile update, the database system 522 checks if the request is an action event in step 1146. If the request is an action event, the database system 522

14

performs the action and replies in step 1148 before returning to step 1122. If the request is not an action event, the database system 522 checks if the request is a security event in step 1150. If the request is not a security event, the database system 522 registers an unknown request type event in step 1152 before returning to step 1122. If the request is a security event, the database system 522 proceeds to step 1154. In step 1154, the database system increments a path/device security record. The database system 522 then appends the requester information to the packet and logs the event before returning to step 1122. In one embodiment of the invention, the database system 522 uses the user access profile system 572 to perform the operations disclosed in FIG. 11 and stores the user access profile in the user access profile system 572.

Service Based Directory

In prior systems, the RADIUS server provided the user with selections for transport modes. However, the user was not able to select available network based services. FIGS. 5 and 12 disclose one embodiment for a service based directory in an example of the invention. In this embodiment, access providers provide a list of services that the user can select. With the list of services, the access providers have the ability to advertise specific services to users. The list of services may be generated based on the user access profile to make the list user specific. Once the user makes the selection, the access server 524 connects the user to the service network such as Intranet, Internet, or private dedicated network that provides the selected service.

FIG. 12 depicts a flowchart for the services based directory in an example of the invention. FIG. 12 begins in step 1200. In step 1202, the access server 524 waits for the next connection from a network device in the user network 510. The access server 524 then receives a connection from the network device 512 in step 1204. Once the connection is established, the access server 524 generates and transmits a user ID query to the network device 512 in step 1206. The access server 524 then receives an ID reply and establishes a network device session in step 1208.

The access server 524 then generates an available services reply including a list of services in 1210. In one embodiment, the access server 524 generates the available services reply based upon information in the user access profile. The access server 524 receives a selected service reply from the network device 512 in step 1212. The access server 524 then connects the network device 512 to the selected service provider in step 1214. The access server 524 waits for the next packet in step 1214. The access server 524 then exchanges packets between the network device 524 and the selected service provider in step 1218.

Access Network User Binding—FIGS. 5, 13, 14, 15, and 16

User Access Profile Mobility

Users may access their user access profile from any network device connected to the network architecture 500. In a prior solution, a command interpreter in an operating system shell environment allows a user to customize the interpretation of command strings. In this environment, the access network identifies the user and the terminal device interface, which provides user mobility. The access network then executes customized actions for the user. FIGS. 5 and 13 show one embodiment of the invention for user access profile mobility. This embodiment provides user mobility in a distributed data network environment.

FIG. 13 depicts a flowchart of user access profile mobility in an example of the invention. FIG. 13 begins at step 1300. A user at the network device 512 signs on the access network

15

520 with their user ID. The network device 512 transmits user information with the user ID to the access server 524. In this embodiment, the user information is in the form of a packet. In step 1302, the access server 524 receives and processes the packet to check if the user access profile is local within the local database system 570. A user access profile is local within the local database system 570 when the user access profile is located in the local database system 570. If the user access profile is local, the access server 524 proceeds to step 1312.

If the user access profile is not local within the local database system 570, the access server 524 checks if the user ID is delimited with a provider ID in step 1304. One example of a user ID delimited with a provider ID includes a user's name and a provider ID separated by a delimiter such as joesmith@access.net. If the user ID is not delimited, the access server 524 retrieves the location of the default user access profile system using a default Lightweight Directory Access Protocol (LDAP) interface system 571 in step 1308 before proceeding to step 1312.

If the user ID is delimited, the access server 524 checks if the provider ID is valid in step 1306. If the provider ID is not valid, the access server 524 registers an invalid profile request security event before terminating at step 1326. If the provider ID is valid, the access server 524 uses the provider ID to retrieve the location of the local database system 590 from a foreign LDAP interface system 591 before proceeding to step 1312.

In step 1312, the access server 524 checks if the packet is a retrieve request for the user access profile. If the packet is a retrieve request, the access server 524 generates and transmits a request to retrieve the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then creates the user session ID and updates the user entries of the user access profile with the user ID and date/time updated in step 1312. The access server 524 then transmits a reply message with the session ID and the location of the LDAP server in step 1314 before terminating at step 1326.

If the packet is not a retrieve request, the access server 524 checks if the packet is a release request in step 1316. If the packet is a release request, the access server 524 retrieves the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then updates the user session ID and updates the user entries of the user access profile with the user ID and date/time updated in step 1318. The access server 524 then transmits a reply message with the session ID in step 1320 before terminating at step 1326.

If the packet is not a release request, the access server 524 checks if the packet is an update request in step 1322. If the packet is not an update request, the access server 524 registers an invalid profile request security event before terminating at step 1326. If the packet is an update request, the access server 524 retrieves the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then updates the user access profile with the information in the packet in step 1324. The access server 524 then transmits a complete message to the user network 510 to signify the profile update is complete before terminating at step 1326. In one embodiment, the local database system 570 uses the user access profile system 572 to retrieve and store the local user access profiles and the local database system 590 uses the user access

16

profile system 592 to retrieve and store the foreign user access profiles.

User, Device, and Service Sessions

Access network providers provide user and device sessions in addition to service sessions to distinguish users when providing communication services. A user session is the information associated with a user accessing a network. A device session is the information associated with a device being used to access a network. A service session is the information associated with a service being provided over a network. Service providers distinguish users instead of access devices or links. This allows multiple users to share a single access device with each user receiving their own customized or preferred services. Advantageously, service providers establish service access rights and restrictions such as preventing adult content for younger viewers or sharing an access device for business and personal use. Also, user, device, and service sessions allow a service provider to group multiple service providers to provide a composite of services to the user similar to a contractor/sub-contractor relationship.

FIGS. 5, 14, 15, and 16 disclose one embodiment for device, user, and service sessions in an example of the invention. FIG. 14 depicts a logical view of device, user, and service sessions in an example of the invention. Devices 1402, 1404, and 1406 are comprised of session type specific information and session links. Session type specific information include public keys, private keys, and session ID. Session links include user sessions, device sessions, and service sessions. The public keys and private keys are for encryption and decryption of messages. Session ID identifies the session ID for the device. User sessions are user sessions that the device is logically linked to. Service sessions are the service sessions the device is logically linked to. Users 1408, 1410, and 1412 are comprised of public keys, private keys, session ID, device sessions, and service sessions. Session ID identifies the session ID for the user. Device sessions are the device sessions the user is logically linked to. Service sessions are the service sessions the user is logically linked to. Services 1414, 1416, and 1418 are comprised of public keys, private keys, session ID, user sessions, device sessions, and sub service sessions. Session ID identifies the session ID for the service. User sessions are the user sessions the service is logically linked to. Device sessions are the device sessions the service is logically linked to. Sub service sessions are the service sessions the service is logically linked to.

In one example, device 1402 is linked to user B 1410 via a link 1422. User B is also linked to service N 1418 via a link 1424. Device 1402 contains the user information in the user session fields for user B 1410 and the service information in the service session fields for service N. User B 1410 contains the device information in the device session fields for device 1402 and the service information in the service session fields for service N 1418. Service N 1418 contains the device information in the device session fields for device 1402 and the user information in the user session fields for user B 1410. Service N 1418 is also linked to service A 1414 via a link 1426. Service N 1418 contains additional service information in the service session fields for service A 1414. Service A 1414 contains the composite service information in the service session fields for service N 1418. Service A 1414 also includes an owning service which is a reference to the service that owns service A.

FIG. 15 depicts a flowchart for a user based session in an example of the invention. FIG. 15 begins in step 1500. In step 1502, the network device 512 establishes a connection

17

with the access server 524. The access server 524 generates and transmits a user ID query to the user network 510 in step 1504. In step 1506, the network device 512 receives the user ID query and transmits a user ID reply to the access server 524. The access server 524 receives and processes the user ID reply to generate a device session. The network device 512 then transmits a packet to the access server 524. The access server 524 then receives the packet in step 1508. In step 1510, the access server 524 processes the packet to check if the packet includes a user session ID.

If the packet does not include the user session ID, the access server 524 transmits a login query encrypted by the network device's 512 public encryption key 1402 to the network device 512 in step 1512. The network device 512 decodes the login query with its public encryption key and transmits a login reply encrypted with its public encryption key in step 1516. The access server 524 then decodes the login reply with the private encryption key and checks if the user ID is valid. If the user is valid, the access server 524 generates a user session and a session ID in step 1520. In some embodiments, the user session ID is the original IP address. The access server 524 then encrypts with the public encryption key and transmits a complete reply with the user session ID to the network device 512 in step 1522 before returning to step 1508.

If the packet does include the user session ID, the access server 524 checks if the user and user session ID are valid in step 1514. If the user or user session ID is not valid, the access server 524 discards the packet and registers a security event in step 1515 before returning to step 1508. If the user and user session ID are valid, the access server 524 generates a service request with the user session ID and the service session ID if available. The access server 524 encrypts the service request with the public encryption key where appropriate. The access server 524 transmits the packet including the service request in step 1518 before returning to step 1508.

FIG. 16 depicts a flowchart for a service based session in an example of the invention. FIG. 16 begins in step 1600. The network device 512 transmits a service request to the access server 524. The access server 524 then receives the service request in step 1602. In step 1604, the access server 524 checks if the service request includes a service session ID.

If the service request does not include the service session ID, the access server 524 transmits a service ID query encrypted with the public encryption key to the network device 512 in step 1606. The network device 512 decodes the service ID query with its private key and transmits a service reply encrypted with the service public encryption key to the access server 524 in step 1610. The access server 524 then decodes the service reply with the service private encryption key and checks if the user is valid. If the user is valid, the access server 524 generates a service session and a session ID in step 1614. The access server 524 then transmits a complete reply with the service session ID to the network device 512 in step 1616 before returning to step 1602.

If the packet does include the service session ID, the access server 524 checks if the user and service session ID are valid in step 1608. In one embodiment, the service session ID is the destination IP address. If the user or service session ID is not valid, the access server 524 discards the packet and registers a security event before returning to step 1602. If the user and service session ID is valid, the access server 524 updates the service request with the user session ID and the service session ID. The access server 524

18

encrypts the service request with the service public encryption key where appropriate. The access server 524 transmits the service request with the user session ID and the service session ID in step 1612 before returning to step 1602.

Access Network Security—FIGS. 5, 17, and 18 Service Capability Firewall

A network service provider interfaces with a network access provider at a transport functional level. The interface typically includes Internet protocol firewalls to provide security. Unfortunately, the interface between the network service provider and the network access provider is not able to hide the implementation details such as addressing schemes, transport details, and equipment specifics. The hiding of implementation details is preferred to prevent hackers from manipulating the implementation details. FIGS. 5, 17 and 18 depict one embodiment for a service capability firewall in an example of the invention. In this embodiment, the interface between the network service provider and the network access provider occurs at a named functional level instead of the transport addressing level. A named function request is any request for a capability of service provided by the network service provider. This allows the network service provider to hide the implementation details. The network service provider exposes only functional capabilities to the users depending on their security rights.

FIG. 17 depicts a flowchart for the firewall 556 for service capability firewall in an example of the invention. FIG. 17 begins in step 1700. In step 1702, the firewall 556 waits for information. In this embodiment, the information is in the form of a packet. The firewall 556 receives and processes a packet including a named function request from the network device 512 in step 1704. The firewall 556 then checks whether the firewall 556 is the destination for the named function request in step 1706. If the firewall 556 is not the destination, the firewall 556 registers a network request security event in step 1710. In step 1712, the firewall 556 encapsulates the packet with path information and transmits the packet to the database system 522 before returning to step 1702.

If the firewall 556 is the destination, the firewall 556 checks whether the sending address is consistent with the path in step 1714. If the sending address is not consistent with the path, the firewall 556 returns to step 1710. If the sending address is consistent with the path and does not belong to the accessing network, the firewall 556 checks if the sending address/session ID/named function combination is cached in step 1716. If the sending address/session ID/named function combination is cached, the firewall 556 replaces the packet's named function request with the private cached destination address in step 1718. The firewall 556 then checks if the private destination address is known and allowed to pass in step 1720. If the destination is known and allowed to pass, the firewall 556 transmits the packet on the correct path with standard firewall access in step 1722 before returning to step 3802. If the destination is not known or not allowed to pass, the firewall 556 returns to step 1710.

If the sending address/session ID/named function combination is not cached, the firewall 556 set an external request timer in step 1724. The firewall 556 then generates and transmits a request for the database system 522 in step 1726. The firewall 556 checks whether a reply is received or the timer has expired in step 1728. If the reply is not received and the timer has not expired, the firewall 556 returns to step 1728. In another embodiment, a blocked I/O is used instead of the wait loop in step 1728. If the reply is received or the timer has expired, the firewall 556 checks if there is a valid

19

reply in step 1730. If the reply is invalid or no reply was received, the firewall 556 discards the packet and registers a translation failure event in step 1734 before returning to step 1702. If the reply is valid, the firewall 556 replaces the packet's named function request based on the reply and caches the address/session functions returned in step 1732 before returning to step 1720.

FIG. 18 depicts a flowchart for the database system 522 for service capability firewall in an example of the invention. FIG. 18 begins in step 1800. In step 1802, the database system 522 waits for the next request. The database system 522 then receives and processes the request in step 1804. In step 1806, the database system 522 then checks if the access server requester is known. If the access server requester is not known, the database system 522 registers an unknown requester event in step 1808. The database system 522 then appends the reply with the packet and replies to the access server 524 in step 1810. The database system 522 then generates and transmits a decline reply to the access server 524 in step 1812 before returning to step 1802.

If the requester is known, the database system 522 then checks if the request is a capability appeal in step 1814. A capability appeal is any message, signaling or instruction requesting a capability with a related network address in the service provider. If the request is not a capability appeal, the database system 522 then checks if the request is a security event in step 1816. If the request is not a security event, the database system 522 register an unknown request event in step 1817 before returning to step 1810. If the request is a security event, the database system 522 increments a path/device security record and appends the requester information to the request packet and logs the event in step 1822 before returning to step 1802. If the request is a capability appeal, the database system 522 then identifies the user and network device 512 in step 1818. The database system 522 then checks if the access is valid in step 1822. If the access is invalid, the database system 522 returns to step 1822. If the access is valid, the database system 522 retrieves the user and network device profiles in step 1824. The database system 522 then checks whether the capability is valid for the user and session state in step 1826. If the capability is invalid, the database system 522 returns to step 1822. If the capability is valid, the database system 522 generates a session ID, the network address of the capability requested, and functions available and appends the network address, which is only sent to the firewall/router 526 and not to the user, to the reply in step 1830. The database system 522 then transmits the approve reply to the access server 524 in step 1832 before returning to step 1802.

Access Network Service Authorization—

FIGS. 5, 19, 20, 21, 22, 23, and 24

Prepaid Access

Prepaid phone cards are commonly used in PSTN, where the customer pays a prepaid amount that is debited against when the customer makes a call. In data networks, prepaid cards are not currently being used. FIGS. 5 and 19 disclose one embodiment for prepaid access in an example of the invention. In this embodiment, users buy prepaid cards from network providers. When the user requests access to one of many access providers throughout the country, the access provider verifies the prepaid account code before providing the access. A prepaid account code is any number that relates to a user's prepaid account. The prepaid account is debited against for the charges related to the access. Other charges related to the service provided may also be debited against the prepaid account. For example, a user may purchase an item from a website and have the charges debited against the

20

prepaid account. Once the prepaid amount is reached, the access provider terminates the access to the user. In one embodiment, the network provider provides different levels of service such as gold, silver, and bronze. The gold service has guaranteed throughput but higher rates for access, while the bronze service has lower throughput and rates.

FIG. 19 depicts a flowchart for prepaid access in an example of the invention. FIG. 19 begins in step 1900. In step 1902, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 1904. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits information identifying a billing code. A billing code is any number that identifies a user for billing. Two examples of billing codes are prepaid account codes and credit card numbers. In this embodiment, the information identifying a billing code is a response including a prepaid account code to the login query. In other embodiments, the billing code is retrieved from the user access profile. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response including the prepaid account code to determine if the user is known in the local database system 520 in step 1906. A user is known when the user is allowed to use the access network 520. In one embodiment, the user is known in the local database system 520 if there is time/amounts left in the user's prepaid account. In another embodiment, the database system 522 evaluates a positive balance file to determine the remaining time/amount in the user's prepaid account. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 1908 before returning to step 1902.

If the user is not known, the database system 522 checks if there is an appeal server for user authentication in step 1910. In one embodiment, the access server 524 performs the appeal on a decline. If there is no appeal server, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2412 before returning to step 2402. If there is an appeal server, the database system 522 generates an authorization query including the prepaid account code for the central database system 580 in step 1914. The database system 522 checks if the user is known in the central database system 580 in step 1914. In one embodiment, the user is known in the central database system 580 if there is time/amounts left in the user's prepaid account. If the user is known, the database system 522 proceeds to step 1908. If the user is not known, the database system 522 proceeds to step 1910.

In one embodiment, the database system 522 uses a user authorization system 575 for checking if the user is known in the local database system 520. The user authorization system 575 contains all the prepaid customers, prepaid customer information, prepaid account codes, and the amount/quantity remaining in the prepaid account to verify if access is allowed. In one embodiment, the database system 522 uses a user authorization system 581 as the appeal server for checking if the user is known in the local database system 580. The user authorization system 581 contains all the prepaid customers, prepaid customer information, and the amount/quantity remaining in the prepaid account.

Bank Card Access

In prior systems, access providers authenticated users using their own databases. FIGS. 5, 20 and 21 disclose one

embodiment for bank card access in an example of the invention. In this embodiment, access providers authenticate users through bank card financial networks using the users' credit card numbers. Network providers use credit or debit card numbers as user ID and passwords for authentication and authorization purposes. Users use prepaid cards, phone cards, and credit cards in PSTN. However, no bank cards have been used for access to data networks other than for batch bill payment.

FIG. 20 depicts a flowchart for bank card access for a connection in an example of the invention. FIG. 20 begins in step 2000. In step 2002, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2004. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits information identifying a billing code. In this embodiment, the information identifying a billing code is a response including credit card numbers to the login query. In other embodiments, the billing code is retrieved from the user access profile. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response to see if the user is known in the local database system 570 in step 2006. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 2008 before returning to step 2002.

If the user is not known, the database system 522 processes the response to check if the user is identified by credit card numbers in step 2010. If the user is not identified by the credit card numbers, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2012 before returning to step 2002. If the user is identified by the credit card numbers, the database system 522 generates an authorization query as a pre-authorization hold or authorization/capture transaction for the central database system 580 in step 2014. In one embodiment, the central database system 580 uses a financial interface 582 to interface with a financial switch to banks for authentication and authorization. The database system 522 then checks if the user is authenticated and authorized by the central database system 580 in step 2016. If the user is authenticated and authorized, the database system 522 logs access information and an authorizing financial entity or institution in step 2018 before proceeding to step 2008. If the user is not known, the database system 522 checks if there is another database system for authorization such as for foreign user access in step 2018. If there is another database system, the database system 522 returns to step 2014. If there is not another database system, the database system 522 returns to step 2012.

FIG. 21 depicts a flowchart for bank card access for a disconnection in an example of the invention. FIG. 21 begins in step 2100. In step 2102, the access server 524 waits for a disconnection. The network device 512 then disconnects from the access server 524 in step 2104. The access server 524 then checks whether the user was on a pre-authorization hold in step 2106. If the user is not on a pre-authorization hold, the 522 logs access information and an authorizing database system 522 in step 2108 before proceeding to step 2102. If the user is on a pre-authorization hold, the database system 522 generates a pre-authorization complete transaction for the central database system 580 in step 2110 before returning to step 2108.

Access Cards

In prior systems, access providers authenticated users using their own databases. FIG. 5, 22 and 23 disclose one embodiment for network access cards in an example of the invention. An access card is a card that a user uses to access a network. The access card includes an access card account code. The access card account code is any number that relates to the user's access account. In this embodiment, access providers authenticate users who have access cards using other access providers' databases. Access providers then may enter into sharing agreements with other access providers. These sharing agreements allow users to use multiple access providers, which provide the users greater mobility and availability. Users use phone cards in PSTN for phone card access. However, no access cards for data networks have been used.

FIG. 22 depicts a flowchart for network access cards for a connection in an example of the invention. FIG. 22 begins in step 2200. In step 2202, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2204. In another embodiment of the invention, the user calls a toll free number to request access. A service control point is queried to determine where to route the request for access similar to an automatic call distribution (ACD). The request for access is then routed to the access server 524 to establish a connection between the network device 512 and the access server 524.

Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits a response including the access card account code to the login query. The access server 524 then transfers the response from the network device 512 to the database system 522. In this embodiment, the database system 522 checks if the user is allowed to use the access network 520 by checking whether the user is known. A user is known when the user is allowed to use the access network 520. For example, a user is known when their user access profile is located within the local database system 570 and their user access profile allows the user to use the access network 520. The database system 522 receives and processes the response including the access card account code to check if the user is known in the local database system 570 in step 2206. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 2208 before returning to step 2202.

If the user is not known, the database system 522 checks if the response included foreign network account information in step 2210. Foreign network account information is any information that is indicative of an account that is external to local database system 570 that the user is attempting to gain access. If there is no foreign network account information, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2212 before returning to step 2202.

If there is foreign network account information, the database system 522 identifies the local database system 590 based on the foreign network account information and generates an authorization query for the local database system 590 in step 2214. The database system 522 then checks if the user is authenticated and authorized by the local database system 590 in step 2216. If the user is authenticated and authorized, the database system 522 logs contract and settlements information returned by the local database system 590 or indicated by the database system 522 in relation to local database system 590 in step 2218 before proceeding to step 2208. If the user is not known, the

23

database system 522 proceeds to step 2212. In one embodiment, the local database system 570 uses the user authorization system 575 to check if the user is known in the local database system 570. In one embodiment, the local database system 590 uses the user authentication system 593 for authentication and authorization.

FIG. 23 depicts a flowchart for network access cards for a disconnection in an example of the invention. FIG. 23 begins in step 2300. In step 2302, the access server 524 waits for a disconnection. The network device 512 then disconnects from the access server 524 in step 2304. The access server 524 then generates and transmits a logoff query for the database system 522. The database system 522 then transfers the logoff query to the local database system 570 and the local database system 590. The database system 522 logs the access information and the authorizing database system in step 2308 before returning to step 2302.

Global Authentication

In prior systems, access providers authenticated users using their own databases. FIGS. 5 and 23 disclose one embodiment for global authentication in an example of the invention. In this embodiment, access providers authenticate users using a centralized database. Access providers then may enter into sharing agreements with other access providers. These sharing agreements allow users to use multiple access providers, which provide the users greater mobility. Currently, cellular telephone companies enter into these sharing arrangements for greater coverage. However, this sharing of databases for authentication and authorization has not occurred for data networks.

FIG. 24 depicts a flowchart for global access in an example of the invention. FIG. 24 begins in step 2400. In step 2402, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2404. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits a response to the login query. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response to see if the user is native in the local database system 570 in step 2406. In this embodiment, the database system 522 checks if the user is allowed to use the access network 520 by checking whether the user is native. A user is native when the user is allowed to use the access network 520. For example, a user is native when their user access profile is located within the local database system 570 and their user access profile allows the user to use the access network 520. If the user is native, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 4308 before returning to step 2402.

If the user is not native, the database system 522 checks if there is an authentication/authorization server in the database system 522 for a foreign network for user authentication in step 1910. If there is no authentication/authorization server in the database system 522 for the foreign network, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2412 before returning to step 2402. If there is an authentication/authorization server in the database system 522 for the foreign network, the database system 522 generates an authorization query for the central database system 580 in step 2414. The database system 522 then checks if the user is known in the central database system 580 in step 2414. If the user is known, the database

24

system 522 proceeds to step 2408. If the user is not known, the database system 522 proceeds to step 2410. In one embodiment, the central database system 580 uses a user authorization system 581 to check if the user is known.

Access Network Proxy/Environment—
FIGS. 5, 25, 26, 27, 28, 29, 30, and 31
User Based Proxy

A proxy is an application that represents itself as one or more network endpoints. The proxy receives a request for services from a user at a network endpoint and acts on behalf of the user in transmitting and receiving user requests and replies. There are Internet proxy agents that are user network access bind points. However, the Internet proxy agents are not user specific, and they act to protect user interests, not network. These proxy agents provide address translation and basic firewall functionality. Client focused proxies have extended to cookie collection and password handling.

FIGS. 5, 25 and 26 disclose one embodiment for user based proxies in an example of the invention. The user based proxies obtain information for the user and establishes a user specific network presence. A benefit of having a user specific network presence is that user access is handled by a process owned by a network security certificate authority that prevents Trojan horse network attacks. User based proxy provides a single control and monitor point for a user. The proxy agents provides a bind point for all user specific access such as user profile functionality, translations, security, and caching.

FIG. 25 depicts a flowchart for the access server 524 for user based proxies in an example of the invention. FIG. 25 begins in step 2500. In step 2502, the access server 524 waits for the next event. If the next event is a connection, the network device 512 connects with the access server 524 in step 2504. In step 2506, the access server 524 then collects user information and transfers the user information to the database system 522 to perform the security. In one embodiment, the database system 522 uses a security server 576 to perform the security. The access server 524 then checks if the user is allowed access in step 2508. If the user is not allowed access, the access server 524 disconnects the user in step 2510 before returning to step 2502.

If the user is allowed access, the access server 524 checks if a proxy is available in the database system 522 in step 2512. In one embodiment of the invention, the user proxies are in the host system 575. If the proxy is available, the access server 524 proceeds to step 2516. If the proxy is not available, the access server 524 starts the proxy in the database system 522 in step 2514 before proceeding to step 2516.

The access server 524 checks if the user access profile information is available in step 2516. If the user access profile information is not available, the access server 524 generates and transmits a user profile error to the network device 512 in step 2518 before returning to step 2502. If the user profile information is available, the access server 524 configures the proxy for the user in step 2520. The access server 524 then generates and transmits a message with the address of the user proxy and the public encryption key to the network device 512 in step 2522. The access server 524 also collects and transmits the access address of the network device 512 and the public encryption key to the user proxy. The access server 524 then returns to step 2502.

If the next event is a disconnection, the network device 512 disconnects with the access server in step 2524. The access server 524 then generates and transmits a reset command to the user proxy to clear the proxy state information and configuration in step 2526 before returning to

25

step 2502. If the next event is a status/request event from the user proxy, the access server 524 sets a user proxy reply timer in step 2528. The user proxy has written a status to the access server 524, and the user proxy receives a reply in step 2530 before returning to step 2502. If the next event is a timer expiration, the access server 524 receives the user proxy reply timer expiration in step 2532. The access server 524 then generates and transmits a continue wait message and status to the user proxy in step 2534 before returning to step 2528.

FIG. 26 depicts a flowchart for a user proxy for user based proxies in an example of the invention. FIG. 26 begins in step 2600. In step 2602, the user proxy waits for the next event. If the next event is the completion of the initialization, the user proxy checks if the initialization is complete in step 2604. The user proxy then sets a request timer in step 2606. The user proxy then request work from the access server 524 and transmits a status to the access server 524 in step 2608 before returning to step 2602. If the next event is the expiration of the request timer, the user proxy checks if the request timer expired in step 2610. The user proxy then registers a timeout in step 2612 before returning to step 2606.

If the next event is an access server 524 reply, the user proxy receives the access server 524 reply in step 2614. The user proxy then checks if the reply is a continue in step 2616. If the reply is a continue, the user proxy returns to step 2606. If the reply is not a continue, the user proxy processes the configuration information or action in step 2618 before returning to step 2606. If the next event is user request, the user proxy receives the user request for service in step 2620. The user proxy then checks if the user is valid in step 2622. If the user is valid, the user proxy then checks if the request is valid in step 2624. If the request is valid, the user proxy packages the request with the security certification and encryption in step 2626. The user proxy then transmits the request to the appropriate network destination in step 2628 before returning to step 2602. If the user or request is not valid, the user proxy registers a security event in step 2630 before returning to step 2602. The user proxy exchanges user information between the user network 510 and the appropriate network destination.

Subscriber Based Proxy

Another type of proxy is a subscriber based proxy. A subscriber is a logical entity such as an organization, corporation, or other grouping of users that has subscribed to services with a service provider. FIGS. 5, 27 and 28 disclose one embodiment for subscriber based proxies in an example of the invention. The subscriber based proxies obtain information for a user of a subscriber group and establishes a subscriber specific network presence. The benefit of having a subscriber specific network presence is that user access rights of the subscriber can be handled as a group, and group rights are owned by a network security certificate authority. The subscriber proxy agents provide a bind point for all subscriber specific access such as user profile functionality, translations, security, and caching.

FIG. 27 depicts a flowchart for the access server 524 for subscriber based proxies in an example of the invention. FIG. 27 begins in step 2700. In step 2702, the access server 524 waits for the next event. If the next event is a connection, the network device 512 connects with the access server 524 in step 2704. In step 2706, the access server 524 then collects user information and transfers the user information to the database system 522 to perform the security. In one embodiment, the database system 522 uses a security server 574 to perform the security. The access server 524

26

then checks if the user is allowed access in step 2708. If the user is not allowed access, the access server 524 disconnects the user in step 2710 before returning to step 2702. If the user is allowed access, the access server 524 checks if all required subscriber proxies are available in the database system 522 in step 2712. In one embodiment of the invention, the subscriber proxies are in the host system 575. If the proxies are available, the access server 524 proceeds to step 2716. If the proxies are not available, the access server 524 starts the required proxies in the database system 522 in step 2714 before proceeding to step 2716.

In step 2716, the access server 524 checks if the subscriber access profile information is available. If the subscriber access profile information is not available, the access server 524 generates and transmits a subscriber profile error to the network device 512 in step 2718 before returning to step 2702. If the subscriber profile information is available, the access server 524 configures the subscriber proxies for the subscriber in step 2720. The access server 524 then generates and transmits a message with the address of the subscriber proxy and the public encryption key to the network device 512 in step 2722. The access server 524 also collects and transmits the access address of the network device 512 and the public encryption key to the subscriber proxies. The access server 524 then returns to step 2702.

If the next event is a disconnection, the network device 512 disconnects with the access server in step 2724. The access server 524 then generates and transmits a reset command to the subscriber proxy in step 2726 before returning to step 2702. If the next event is a status/request event from the subscriber proxy, the access server 524 sets a subscriber proxy reply timer in step 2728. In step 2730, the subscriber proxy has written a status to the access server 524, and the subscriber proxy receives a reply before returning to step 2702. If the next event is a subscriber proxy timer expiration, the access server 524 receives the subscriber proxy reply timer expiration in step 2732. The access server 524 then generates and transmits a continue wait message and a status to the subscriber proxy in step 2734 before returning to step 2730.

FIG. 28 depicts a flowchart for a subscriber proxy for subscriber based proxies in an example of the invention. FIG. 28 begins in step 2800. In step 2802, the subscriber proxy waits for the next event. If the next event is the completion of the initialization, the subscriber proxy checks if the initialization is complete in step 2804. The subscriber proxy then sets a request timer in step 2806. The subscriber proxy then request work from the access server 524 and transmits a status to the access server 524 in step 2808 before returning to step 2802. If the next event is the expiration of the request timer, the subscriber proxy checks if the request timer expired in step 2810. The subscriber proxy then registers a timeout in step 2812 before returning to step 2806.

If the next event is an access server 524 reply, the subscriber proxy receives the access server 524 reply in step 2814. The subscriber proxy then checks if the reply is a continue in step 2816. If the reply is a continue, the subscriber proxy returns to step 2806. If the reply is not a continue, the subscriber proxy processes the configuration information or action in step 2818 before returning to step 2806. If the next event is a user request for service, the subscriber proxy receives the user request in step 2820. The subscriber proxy then checks if the user is valid in step 2822. If the user is valid, the subscriber proxy then checks if the request is valid in step 2824. If the request is valid, the subscriber proxy packages the request with the security

certification and encryption in step 2826. The subscriber proxy then transmits the request to the appropriate network destination in step 2828 before returning to step 2802. If the user or request is not valid, the subscriber proxy registers a security event in step 2830 before returning to step 2802. The subscriber proxy exchanges user information between the user network 510 and the appropriate network destination.

Dynamic Proxies

One prior system named "pluxy" allows a manual extension of a proxy. Pluxy requires manual determination and loading of a dynamic set of services, which is done on an operator basis. FIGS. 5 and 29 disclose one embodiment for dynamic proxies in an example of the invention. In this embodiment, both the user based proxies and the subscriber based proxies are extended in response to user actions. The dynamic proxy can be modified and enhanced to evolve with new services and/or protocols. The dynamic proxy size is smaller and more efficient by supporting only the logic that is being used. Implementation and development times of new service and protocol are also reduced because new proxies are not required with the new service and/or protocol.

FIG. 29 depicts a flowchart for a dynamic proxy for dynamic proxies in an example of the invention. FIG. 29 begins in step 2900. In step 2902, the dynamic proxy waits for the next event. If the next event is the completion of the initialization, the dynamic proxy checks if the initialization is complete in step 2904. The dynamic proxy then sets a request timer in step 2906. The dynamic proxy then request work from the access server 524 in step 2908 before returning to step 2902. If the next event is the expiration of the request timer, the dynamic proxy checks if the request timer expired in step 2910. The dynamic proxy then registers a timeout in step 2912 before returning to step 2906.

If the next event is an access server 524 reply, the dynamic proxy receives the access server 524 reply in step 2914. The dynamic proxy then checks if the reply is a continue in step 2916. If the reply is a continue, the dynamic proxy returns to step 2906. If the reply is not a continue, the dynamic proxy processes the configuration information in step 2918 before returning to step 2902. If the next event is a user request, the dynamic proxy receives the user request in step 2920. The dynamic proxy then checks if the user and the request are valid in step 2922. If the user and the request are valid, the dynamic proxy checks if there is a new service or protocol requested in step 2924. If there is a new service or protocol, the dynamic proxy proceeds to step 2934. If there is not a new service or protocol, the dynamic proxy packages the request with the security certification and encryption in step 2926. The dynamic proxy then transmits the request to the appropriate network destination in step 2928 before returning to step 2902. If the user or request is not valid, the subscriber proxy registers a security event in step 2930 before returning to step 2902.

If the next event is a new service/protocol request, the dynamic proxy receives the new service/protocol request in step 2932. In step 2934, the dynamic proxy processes the new service/protocol request to generate a handler request for a new service/protocol handler in step 2934 before returning to step 2926. The dynamic proxy receives the handler and executes the handler for the new service or proxy. In one embodiment, the dynamic proxy is enhanced by loading apple-like extensions from a handler system similar to a web browser.

Access Execution Environment

Proxy servers in the Internet represents a user in one network as existing in another network to obtain services for

the user. One problem with Internet proxy servers is the absence of the management of network resource utilization. In TINA-C, the User Agent and the User Session Manager are restricted to known service binding because any new service requires knowledge of or logic for a new CORBA interface. One problem with TINA-C is the absence of any provider resource management capabilities. Unfortunately, both Internet proxy servers and TINA-C do not provide any means of associating or viewing all network resources used by a user access session. Another problem is the lack of securing provider network resource against tampering by accessing users. Also, Internet proxy servers and TINA-C do not provide a platform-supported means of enforcing security levels on network access users. Both system also lack a validation of potential execution capability extensions.

FIGS. 5, 30, and 31 disclose one embodiment for an access execution environment in an example of the invention. The access execution environment easily and efficiently manages and secures network access by providing an execution environment in the access provider environment. Users access shared services and resources through their specific access execution environment. Because the access execution environment is in the access provider environment, the access provider can view the all of the user's activity by observing the execution environment. In this embodiment, the execution environment is setup as a standard system user on the network access platform, which allows management of the system user by operating system level user resource controls, quotas, and management mechanisms.

The access execution environment includes a multi-tasking virtual machine, a script interpreter, alias resolution capabilities, security certificate authentication, configuration handler, session caching, and protocol handling components. FIG. 30 depicts a block diagram of the access execution environment in an example of the invention. In FIG. 30, a network access platform 3000 comprises an execution environment manager 3010, a program and attribute database 3020, an execution environment A 3030, a transport A 3032, an execution environment B 3040, a transport B 3042, an execution environment C 3050, a transport C 3052, an execution environment D 3060, a transport D 3062, an execution environment E 3070, and a transport E 3072. In one embodiment, the network access platform 3000 is included within the database system 522. In another embodiment, the network access platform is included within the host system 575. The transport A 3032 is connected to the execution environment A 3030. The transport B 3042 is connected to the execution environment A 3040 and a workstation 3080. The transport C 3052 is connected to the execution environment C 3050. The transport D 3062 is connected to the execution environment D 3060. The transport E 3072 is connected to the execution environment E 3070. The execution environment A 3030 includes a transport handler 3034, a configuration handler 3036, and a security handler 3038. The execution environment B 3040 includes a transport handler 3044, a configuration handler 3046, and a security handler 3048. The execution environment C 3050 includes a transport handler 3054, a configuration handler 3056, and a security handler 3058. The execution environment D 3060 includes a transport handler 3064, a configuration handler 3066, and a security handler 3068. The execution environment E 3070 includes a transport handler 3074, a configuration handler 3076, and a security handler 3078.

In operation, a system administrator sets up a number of network access user accounts on the hardware platform. The administrator then sets up resource limits for each account/

platform resource. The administrator then starts an execution environment **3030**, **3040**, **3050**, **3060**, and **3070** for each access user account ID. The execution environment **A 3030** initializes and loads a configuration handler **3036**. The execution environment **A 3030** then loads the security handler **3038** and the transport handler **3034**. The transport handler **3034** opens a logical or physical transport port **A 3032** on the hardware platform using port information from the configuration handler.

FIG. 31 depicts a flow chart for the access execution environment in an example of the invention. FIG. 31 begins in step **3100**. In step **3102**, the security handler **3038** waits for a connect message from the transport handler **3034**. The security handler **3038** then receives the connect message from the transport handler **3034** in step **3104**. The security handler **3038** then generates and transmits a logon request via the transport handler **3034** in step **3106**. The security handler **3038** then checks if a reply for the logon request from the transport handler **3034** has been received in step **3108**. If the reply for the logon request has not been received, the security handler **3038** checks if a reply timeout has occurred in step **3110**. If a reply timeout has not occurred, the security handler **3038** returns to step **3108**. If a reply timeout has occurred, the security handler **3038** generates and transmits a disconnect message to the transport handler **3034** in step **3112** and returns to step **3102** to wait for a connect message.

In step **3114**, the security handler **3038** receives the reply for the logon request. Then the security handler **3038** retrieves the location of a security server from the configuration handler **3036** and transmits the logon information to the security server in step **3116**. The security handler **3038** then checks if a reply to the logon information from the security server has been received in step **3118**. If no reply has been received, the security handler **3038** checks if a reply timeout has occurred in step **3120**. If a reply timeout has not occurred, the security handler **3038** returns to step **3118**. If a reply timeout has occurred, the security handler **3038** checks if the logon information has been sent three times in step **3122**. The number of tries could be configurable. If the logon information has not been sent three times, the security handler **3038** returns to step **3116** to transmit the logon information. If the logon information has been sent three times, the security handler **3038** returns to step **3112** to send a disconnect message.

If the security server replied before the reply timeout, the security handler **3038** checks if the reply is an accept message in step **3124**. If the reply is a decline message, the security handler **3038** returns to step **3112**. If the reply is an accept message, the security handler **3038** transmits the accept message configuration parameters to the configuration handler **3036** in step **3126**. The configuration handler **3036** then loads attributes and programs and executes programs specified by the security server reply in step **3128**. The execution environment **A 3030** performs the execution of programs for the user in step **3130**. In another embodiment, the programs request attributes and/or programs to be loaded and/or executed. The programs include the ability to interface with users via the transport handler and load/execute other programs required by request types.

The configuration handler **3036** then checks if the transport handler **3034** receives a disconnect message in step **3132**. If the transport handler **3034** has not received a disconnect message, the configuration handler **3036** returns to step **3130**. If the transport handler **3034** has received a disconnect message, the transport handler **3034** transmits the disconnect message to the configuration handler **3036** in step

3134. In step **3136**, the configuration handler **6536** based on port configuration attributes closes the transport port **3032**, resets the transport handler **3034**, and notifies the execution environment manager **3010** to create a new execution environment for that port. The configuration handler **3036** gracefully shuts down all handlers, programs, and eventually the execution environment **A 3030**. The configuration handler **3036** transmits a shutdown message to the execution environment manager **3010** in step **3138** before terminating in step **3140**. The execution environment manager **3010** restarts the execution environment **A 3030** upon notification of the shutdown. The operations of the other execution environments **3040**, **3050**, **3060**, and **3070** perform in the same manner as the execution environment **A 3030** and are not discussed for the sake of clarity.

Access Network Translations—FIGS. 5, 32, 33, 34, 35, 36, 37, 38, and 39

Domain Name Scoping

The typing in of domain names such as www.nypostonline.com is quite cumbersome for the user to access specific services. The Domain Name Server (DNS) translates the domain name to a network address for the service. Clicking on bookmarks or favorites in browsers, which reference the domain names, does eliminate the need to type in the domain names. However, when users change network devices, these bookmarks or favorites do not follow the user. In an operating system environment, the command interpreter shell allows the user to customize interpretation of command strings. If no customization is loaded, the command interpreters interprets in a default fashion.

FIGS. 5, 32, and 33 disclose one embodiment for domain name scoping in an example of the invention. This embodiment provides a customizable network shell to overlay the standard DNS service offered in Internet Protocol based networks. Users then are able to define user specific acronyms or aliases for specific or logical services. An acronym or alias indicates domain names, macros, programs, actions, or network addresses. If the translation for the alias does not exist in the list of aliases for the user, the access server **524** checks if the alias exists in the list for a group in which the user belongs. The access server **524** then checks if the alias exist in the DNS for the general network. In another embodiment, the database system **522** checks the existence of the alias in the list for the group and in the DNS for the general network. There are numerous variations in the hierarchy for searching for an alias but are not discussed for the sake of simplicity.

FIG. 32 depicts a flowchart for the access server **524** for domain name scoping in an example of the invention. FIG. 32 begins in step **3200**. In step **3202**, the access server **524** waits for a packet. In this embodiment, the access server **524** receives information in the form of packets from the network device **512**. The access server **524** then receives and processes the packet from the network device **512** in step **3204**. The access server **524** then checks if the packet is an alias translation request in step **3206**. If the packet is not an alias translation request, the access server **524** transmits the packet with standard access logic in step **3214** before returning to step **3202**.

If the packet is an alias translation request, the access server **524** then checks if the alias translation is cached for the user in step **3208**. If the alias translation is cached, the access server **524** reformats the packet using the cached alias translation in step **3216** before returning to step **3214**. If the alias translation is not cached, the access server **524** sets an external request timer in step **3210**. The access server **524** then generates and transmits an alias translation request to

the database system 522 in step 3212. The alias translation request is any message, instruction, or signaling indicative of requesting an alias translation. The access server 524 checks whether a reply has been received or the external request timer has expired in step 3218. If a reply has not been received and the external request timer has not expired, the access server 524 returns to step 3218.

If a reply has been received or the external request timer has expired, the access server 524 checks if the reply was valid in step 3220. If the reply is invalid, the access server 524 discards the packet and replies with an invalid alias message or alias error message to the network device 512 before returning to step 3202. If the reply is valid, the access server 524 caches the alias translation for the user in step 3222 before returning to step 3216.

FIG. 33 depicts a flowchart for the database system 522 for domain name scoping in an example of the invention. FIG. 33 begins in step 3300. In step 3302, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 3304. In step 3306, the database system 522 then checks if the access server requester is known in step 3306. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 3308. The database system 522 appends the reply information to the packet in step 3310. The database system 522 replies with a decline message to the access server 524 in step 3312 before returning to step 3302.

If the access server requester is known, the database system 522 then checks whether the request is an alias translation request in step 3314. If the request is not an alias translation request, the database system 522 registers an unknown request event in step 3316 before returning to step 3310. If the request is an alias translation request, the database system 522 identifies and authenticates the user and the network device 512 in step 3318. If the access is invalid, the database system 522 increments a path/device security record in step 3322. The database system 522 then appends the requester denial information to the request packet in step 3324. The database system 522 then replies with a decline message to the access server 524 in step 3312 before returning to step 3302.

If the access is valid, the database system 522 retrieves the user's alias translation list in step 3326. The database system 522 then translates the alias for the user and appends the translation to a reply to the access server 524. The database system 522 then replies with an approve message to the access server 524 in step 3330. In one embodiment of the invention, the database system 522 uses the alias translation system 578 to perform the operations disclosed in FIG. 33 and stores the aliases and alias translations in the alias translation system 578.

Inband Domain Name Service Lookup

The Domain Name Server (DNS) translations of a domain name to a network address delays user interaction with the service provider. In prior solutions in voice telephone Signaling System #7 networks, a signal transfer point performs an inband local number portability (LNP) lookup on a ISDN part call setup request such as an Initial Address Message (IAM). This inband lookup eliminates the network switch from launching a Transaction Capabilities Application Part (TCAP) LNP query to translate the telephone number in the IAM.

FIGS. 5 and 34 disclose one embodiment for inband domain name service in an example of the invention. This embodiment provides a cache for the access server 524 for alias translations on a per user basis. Thus, the external

queries for translations of aliases to domain names, macros, programs, or network addresses are eliminated. If the translated value of a request is in the alias translation cache, no translation request will be required to a DNS server. Therefore, users experience reduced delays when requesting a service.

FIG. 34 depicts a flowchart for the access server 524 for an inband domain name service lookup in an example of the invention. FIG. 34 begins in step 3400. In step 3402, the access server 524 waits for a packet. The access server 524 then receives and processes the packet from the network device 512 in step 3404. The access server 524 then checks if the destination network address and user is valid in step 3406. If the destination network address and user is valid, the access server 524 transmits the packet on the correct path to reach the destination in step 3408 before returning to step 3402.

If the destination network address or the user is invalid, the access server 524 then checks if the alias translation is cached for the user in step 3410. If the alias translation is cached, the access server 524 reformats the packet using the cached alias translation in step 3412 before returning to step 3402. If the alias translation is not cached, the access server 524 sets an external request timer in step 3414. The access server 524 then generates and transmits an alias translation request to the database system 522 in step 3416. The access server 524 checks whether a reply has been received or the external request timer has expired in step 3418. If a reply has been received and the external request timer has not expired, the access server 524 returns to step 3418.

If a reply has been received or the external request timer has expired, the access server 524 checks if the reply was valid in step 3420. If the reply is invalid, the access server 524 discards the packet and replies with an invalid alias message to the network device 512 in step 3424 before returning to step 3402. If the reply is valid, the access server 524 caches the alias translation for the user in step 3422 before returning to step 3412.

Inline Access Service Triggering

Typical firewalls filter packets out on a request by request basis on "what is not allowed". FIGS. 5 and 35 disclose one embodiment for inline access service triggering. In this embodiment, the access server 554 filters packets out on a "what is allowed" basis. Also, the access server 554 performs packet filtering on a service access basis.

FIG. 35 depicts a flowchart for inline access service triggering in an example of the invention. FIG. 35 begins in step 3500. In step 3502, the access server 554 waits for the next packet. In this embodiment, the access server 554 receives information in the form of packets from the network device 512. The access server 554 receives and processes a packet from the network device 512 in step 3504. In step 3506, the access server 554 checks if the protocol, sending address, and the destination address are known and allowed to pass. In one embodiment, the access server 554 checks the database system 522 for the determination made in step 3506. If the protocol, sending address, and the destination address are known and allowed to pass, the access server 554 checks if the sending address is consistent with the path in step 3508. If the sending address is not consistent with the path, the access server 554 registers a path security event in step 3510. The access server 554 then formats the packet with path information in step 3512. The access server 554 also increments the path/device security record and logs the event before returning to step 3502. If the sending address is consistent with the path, the access server 554 transmits the packet on the correct path with standard firewall access in step 3514 before returning to step 3502.

If the protocol, sending address, or the destination address are not known or not allowed to pass, the access server 554 checks if the request is a filter appeal in step 3515. If the request is not a filter appeal, the access server 554 discards the packet in step 3526 before returning to step 3502. If the request is a filter appeal, the access server 554 identifies and authenticates the user and the network device 512 in step 3516. The access server 554 then checks if the access is valid in step 3518. If the access is invalid, the access server 554 returns to step 3510. If the access is valid, the access server 554 retrieves the user access profile and network device's 512 profile in step 3520. The access server 554 then modifies the access logic for filter modification in step 3524. The access server 554 then modifies the protocol and address filters based on the request in step 4824 before returning to step 3514. In some embodiments, the access server 554 modifies the filters in conformance with the user access profile.

Access Service Triggering

Access providers sometimes need to extend their authentication logic beyond their primary access devices. No prior system in data networks extends the authentication logic beyond what is performed by the access provider's access devices. FIGS. 5, 36 and 37 disclose one embodiment for access service triggering. In this embodiment, the access server 524 triggers a request to external access control logic. Access providers extend the access and authentication logic beyond their access devices while maintaining centralized control of the authentication logic and user access profiles. Also, the access server 554 performs packet filtering on a service access basis.

FIG. 36 depicts a flowchart for the access server 524 for access service triggering in an example of the invention. FIG. 36 begins in step 3600. In step 3602, the access server 524 waits for the next packet. The access server 524 then receives and processes a packet from the network device 512 in step 3604. In step 3606, the access server 524 checks if the protocol, sending address, and the destination address are known and allowed to pass. In one embodiment, the access server 524 checks the database system 522 for the determination made in step 3606. If the protocol, sending address, and the destination address are known and allowed to pass, the access server 524 checks if the sending address is consistent with the path in step 3608. If the sending address is not consistent with the path, the access server 524 registers a path security event in step 3610. The access server 524 then formats the packet with path information in step 3612 before returning to step 3602. If the sending address is consistent with the path, the access server 524 transmits the packet on the correct path with standard firewall access in step 3614 before returning to step 3602.

If the protocol, sending address, or the destination address are not known or are not allowed to pass, the access server 524 sets an external request timer in step 3616. The access server 524 then generates and transmits a request with the path information to the database system 522 in step 3618. The access server 524 then checks if a reply has been received or the external request timer has expired in step 3620. If the reply has not been received or the external request timer has not expired, the access server 524 checks if the reply is valid in step 3622. If the reply is invalid, the access server 524 discards the packet in step 3624 before returning to step 3602. If the reply is valid, the access server 524 then modifies the protocol and address filters based on the reply in step 3624 before returning to step 3614.

FIG. 37 depicts a flowchart for the database system 522 for access service triggering in an example of the invention.

FIG. 37 begins in step 3700. In step 3702, the database system 522 waits for the next request. The database system 522 then receives and processes the request in step 3704. In step 3706, the database system 522 then checks if the access server requester is known in step 3706. If the access server requester is not known, the database system 522 registers an unknown requester event in step 3708. The database system 522 then appends the reply to the packet in step 3710. The database system 522 then generates and transmits a decline reply to the access server 524 in step 3712 before returning to step 3702.

If the access server requester is known, the database system 522 then checks if the request is a filter appeal in step 3714. If the request is not a filter appeal, the database system 522 then checks if the request is a security event in step 3716. If the request is not a security event, the database system 522 registers an unknown action event in step 3717 before returning to step 3712. If the request is a security event, the database system 522 increments a path/device security record, appends the requester information to the request packet, and logs the event in step 3718. The database system 522 then checks if the request is a security event in step 3719. If the request is not a security event, the database system 522 then returns to step 3712. If the request is a security event, the database system 522 proceeds to step 3728.

If the request is a filter appeal, the database system 522 then identifies the user and network device 512 in step 3720. The database system 522 then checks if the access is valid in step 3722. If the access is invalid, the database system 522 returns to step 3718. If the access is valid, the database system 522 retrieves the user and network device profiles in step 3724. The database system 522 then generates access logic and appends the access logic to a reply in step 3726. The database system 522 then transmits the approve reply to the access server 524 in step 3728 before returning to step 3702.

Personal URL

The Internet currently uses Uniform Resource Locator (URL) addresses such as www.yahoo.com so that the address is in more human readable form than the network address. Unfortunately, user cannot distinguish themselves by a network address because the user's network address changes when the user's access point changes. FIGS. 5, 38 and 39 disclose one embodiment for a personal URL. In this embodiment, a network user may publish their location on the network by a user alias. A user alias is any alias that relates to the network address of a user. Logically linking a user's current network address with a user alias allows a user to be located wherever the user accesses the network.

FIG. 38 depicts a flowchart for a personal URL lookup in an example of the invention. FIG. 38 begins in step 3800. In step 3802, the database system 522 waits for the next packet. The database system 522 then receives and processes the packet including the user alias. The database system 522 then checks if a user alias translation is cached for this user in step 3806. If the user alias translation is cached, the database system 522 replies with the current network address of the user in step 3808 before returning to step 3802.

If the user alias translation is not cached, the database system 522 sets an external request timer in step 3810. The database system 522 then generates and transmits a request for user alias translation logic in step 3812. In one embodiment, the database system 522 transmits the request for user alias translation logic to the database system that contains the user access profile. The database system 522

then checks whether a reply was received or the external request timer expired in step 3814. If the reply was not received and the external request timer did not expire, the database system 522 returns to step 3814. If a reply was received or the external request timer did expire, the database system 522 checks if the reply was valid in step 3816. If the reply was invalid, the database system replies with an invalid name message in step 3824 before returning to step 3802.

If the reply was valid, the database system 522 check if there is a current network address for the user in step 3818. If there is no current network address for the user, the database system 522 replies with user alias not currently in network message in step 3820 before returning to step 3802. If there is a current network address for the user, the database system 522 caches the user alias translation in step 3822 before returning to step 3808.

FIG. 39 depicts a flowchart for a personal URL update in an example of the invention. FIG. 39 begins in step 3900. In step 3902, the database system 522 wait for a request. The database system 522 then receives and processes the request to update the user alias in step 3904. In one embodiment, the database system receives the request from a database system that contains the user access profile. In another embodiment, the request comes from the access server 524. The database system 522 then checks if the requester is known in step 3906. If the requester is not known, the database system 522 registers an unknown requester event in step 3910. The database system 522 then transmits a reply with decline in step 3912 before returning to step 3902.

If the requester is known, the database system 522 then checks if the requester is allowed to update the user alias in step 3914. If the requester is not allowed, the database system 522 registers an unauthorized requester event in step 3916 before returning to step 3912. If the requester is allowed to update, the database system 522 then identifies and authenticates the user and current network address in step 3918. The database system 522 then checks if the access is valid in step 3920. If the access is invalid, the database system 522 increments the path/device security record in step 3922. The database system 522 also appends the requester information to the request and logs the event before returning to step 3912.

If the access is valid, the database system 522 then updates the user alias translation with the current network address in step 3924. The database system 522 then replies with an approve in step 3926 before returning to step 3902. In one embodiment of the invention, the database system 522 uses the personal DNS system 579 to perform the operations disclosed in FIGS. 38 and 39.

Access Network Caching—FIGS. 5, 40, 41, 42, 43, and 44

Predictive Caching

Users that are dialed into the network at a rate of 56 kbps or greater typically retrieve information at a rate of 2–4 kbps. Various equipment between the network access provider and the service provider cause this lower rate of transmission. Some examples of the equipment are network access provider equipment, service provider equipment, network backbone overloading, traffic shaping device, firewalls, and routers. One solution for compensating for the lower rate of transmission is caching. Computers typically contain a memory cache for specific application or devices. Firewalls and routers may also contain caches for data. Unfortunately, none of these caches are user specific.

FIGS. 5, 40, 41 and 42 disclose one embodiment of the invention for predictive end user caching. Predictive end

user caching advantageously improves user noticeable delays and slowdowns due to the lower transmission rate. The extension of data requests from the user network 510 beyond the access server 524 is eliminated when the data requested is cached in the database system. Also, using a predictive algorithm reduces delay by improving caching efficiency based on a user's predictable pattern.

FIG. 40 depicts a flowchart for the access server 524 for auditing in an example of the invention. FIG. 40 begins at step 4000. In step 4002, the user network 510 establishes a connection to the access server 524. In step 4004, the access server 524 generates and transmits a login request message to the database system 522. The access server 524 then checks if the database system allowed an access session to be established for the user in step 4006.

If the access session is not established, the access server 524 terminates in step 4020. If the access session is established, the access server 524 then checks if an access session tear down is requested in step 4012. If the access session tear down is requested, the access server 524 generates and transmits a tear down message with user and path information to the database system 522 in step 4014 before terminating in step 4020.

While no session tear down request is received, the access server 524 exchanges packets with the user network 510 depending on the service provided in step 4016. The access server 524 transmits all new packet destinations including the user and path information to the database system 522 in step 4018 before returning to step 4012. In one embodiment, the database system 522 stores the connect request message, the session reject message, the session accept message, the tear down message including the user and path information, and the packets including the user and path information in an audit database system 526. In one embodiment, the connect request message, the session reject message, the session accept message, the tear down message including the user and path information, and the packets including the user and path information are in the form of an audit message. An audit message is any message, information, or signaling that is audited while a user is accessing an access network 520.

FIG. 41 depicts a flowchart for the database system 522 for auditing in user predictive caching in an example of the invention. FIG. 41 begins in step 4100. In step 4102, the database system 522 waits for an audit message. In step 4104, the database system 522 receives and processes the audit message. The database system 522 then checks if the audit message is a session accept message or a tear down message in step 4106. If the audit message is not a session accept message or a tear down message, the database system 522 retrieves the session state object and updates the object with the audit message in step 4112 before returning to step 4102.

If the audit message is a session accept message or a tear down message, the database system 522 then checks if the audit message is a session accept message in step 4108. If the audit message is a session accept message, the database system 522 generates a session state object including the user, device, path, and session ID in step 4114 before returning to step 1802. If the audit message is a tear down message, the database system 522 retrieves the session state object and updates the object in step 4112. Also, the database system 522 stores the event in step 4116. In one embodiment, the database system 522 stores the event in the cache database system 574. The database system 522 removes the active reference from the session state object before returning to step 4102. In one embodiment of the invention, the database system 522 uses the audit database system 573 to perform the operations disclosed in FIG. 41.

FIG. 42 depicts a flowchart for the database system 522 for caching in user predictive caching in an example of the invention. FIG. 42 begins in step 4200. In step 4202, the database system 522 waits for the next event. The database system 522 processes the event in step 4204. In this embodiment, a request for data is a cache request. The database system 522 then checks if the event is a cache request in step 4206.

If the event is a cache request, the database system 522 identifies the user and retrieves the cached user data in step 4208. The database system 522 then checks if the requested data from the cache request is in the cached user data in step 4212. If the requested data is in the cached user data, the database system 522 replies with the requested data in step 4210 before returning to step 4202. If the requested data is not in the cached user data, the database system 522 retrieves the requested data from the service provider via the access server 524 in step 4214. The database system 524 then replies to the user network 510 with the requested data. Alternatively, in another embodiment, the access server 524 transfers the requested data from the service provider to the user network 510 and the database system 522.

In step 4222, the database system 522 updates the user access profile and the caching algorithm. The database system 522 then determines whether there is a change in the cached data based on the caching algorithm in step 4230. If there is no change in the cached data, the database system 522 returns to step 4202. If there is a change in the cached data, the database system 522 sets any new caching timers for the cached data in step 4236. The database system 522 also generates and transmits a translation change to the access server 524 to change the requested destination address to the database system 522 in step 4238 before returning to step 4202.

If the event is not a cache request, the database system 522 then checks if the event is an audit or timer request in step 4216. If the event is an audit or timer request, the database system 522 identifies the user and retrieves the user access profile in step 4218. The database system 522 then checks if the event is an audit request in step 4220. If the event is an audit request, the database system 522 proceeds to step 4222. If the event is not an audit request, the database system 522 refreshes the cached data from the service provider of the cached data based on the caching algorithm in step 4228. The database system 522 then resets the caching timer for the cached data in step 4234 before returning to step 4202.

If the event is not an audit or timer request, the database system 522 checks if the event is a reset event in step 4224. If the event is not a reset event, the database system 522 registers a unknown event received in step 4232 before returning to step 4202. If the event is a reset event, the database system 522 clears the user access profile statistics, caching timers, the user's cached data. In one embodiment, the user's cached data is stored in the cached database system 574 and the database system 522 uses the cached database system 574 to perform the operations disclosed in FIG. 42.

User Controlled Caching

Another solution to reduce user noticeable delays and slowdowns due to the lower transmission rate is user controlled caching. FIGS. 5, 40, 43 and 44 disclose one embodiment of the invention for user controlled caching. A user controls caching by creating a script of commands to cache data prior to the user accessing the network. For example, a user selects financial or local weather forecasts to cache, so when the user logs on to the network the information will be immediately available from the cache. Scripting has typi-

cally been done on general purpose computers. In a general purpose computer, the user sets up a sequenced set of commands to be executed when the script is executed. Unfortunately, this type of scripting has not been performed on a network cache.

In this embodiment of user controlled caching, the operation of the access server 524 is as described in FIG. 40. FIG. 43 depicts a flowchart for the database system 522 for auditing in user controlled caching in an example of the invention. FIG. 43 begins in step 4300. In step 4302, the database system 522 waits for an audit message. In step 4302, the database system 522 receives and processes the audit message. The database system 522 then checks if the audit message is a session accept message or a tear down message in step 4306. If the audit message is not a session accept message or a tear down message, the database system 522 retrieves the session state object and updates the object with the audit message in the cache in step 4312 before returning to step 4302.

If the audit message is a session accept message or a tear down message, the database system 522 then checks if the audit message is a session accept message in step 4308. If the audit message is a session accept message, the database system 522 generates a session state object in the cache including the user, device, path, and session ID in step 4314. The database system 522 then checks if the user has a pre-cache script in step 4318. If the user does not have a pre-cache script, the database system 522 returns to step 4302. If the user has a pre-cache script, the database system 522 generates and transmits a pre-cache instruction set to the access server 524 to change the translation filter to access the database system 522 for destinations in the pre-cache script so that any requests for those destinations are fulfilled from the cache in step 4320. The database system 522 then sets cache refresh timers in step 4322 and returns to step 4302.

If the audit message is a tear down message, the database system 522 retrieves the session state object and updates the object in step 4316. Also, the database system 522 stores the event in step 4316. In one embodiment, the database system 522 stores the session state object and the state in the cache database system 527. The database system 522 removes the active reference from the session state object before returning to step 4302. In one embodiment of the invention, the database system 522 uses the audit database system 573 to perform the operations disclosed in FIG. 43.

FIG. 44 depicts a flowchart for the database system 522 for caching in user controlled caching in an example of the invention. FIG. 44 begins in step 4400. In step 4402, the database system 522 waits for the next event. The database system 522 processes the event in step 4404. In this embodiment, a request for data is a cache request. The database system 522 then checks if the event is a cache request in step 4406.

If the event is a cache request, the database system 522 identifies the user and retrieves the cached user data in step 4408. The database system 522 then checks if the requested data from the cache request is in the cached user data in step 4412. If the requested data is in the cached user data, the database system 522 replies with the requested data in step 4410 before returning to step 4402. If the requested data is not in the cached user data, the database system 522 retrieves the requested data from the service provider via the access server 524 in step 4414. The database system 522 then replies to the user network 510 with the requested data. Alternatively, in another embodiment, the access server 524 transfers the requested data from the service provider to the user network 510 and the database system 522.

In step 4422, the database system 522 updates the user access profile and the caching algorithm. The database system 522 then determines whether there is a change in the cached data based on the caching algorithm in step 4430. If there is no change in the cached data, the database system 522 returns to step 4402. If there is a change in the cached data, the database system 522 sets the new caching timer for the cached data in step 4436. The database system 522 also generates and transmits a translation change to the access server 524 to change the requested destination address to the database system 522 in step 4440 before returning to step 4402.

If the event is not a cache request, the database system 522 then checks if the event is an audit or timer request in step 4416. If the event is an audit or timer request, the database system 522 identifies the user and retrieves the user access profile in step 4418. The database system 522 then checks if the event is an audit request in step 4420. If the event is an audit request, the database system 522 proceeds to step 4422.

If the event is not an audit request, the database system 522 checks if the event is a script timer in step 4428. If the event is not a script timer, the database system 522 refreshes the cached data from the service provider of the cached data based on the caching algorithm in step 4438 before returning to step 4402. If the event is a script timer, the database system 522 executes the user commands or command set specified by the script timer event in step 4434. The database system 522 then resets the script timer for the cached data in step 4442 before returning to step 4402.

If the event is not an audit or timer request, the database system 522 checks if the event is a reset event in step 4424. If the event is not a reset event, the database system 522 registers a unknown event received in step 4432 before returning to step 4402. If the event is a reset event, the database system 522 clears the user access profile statistics, caching timers, the user's cached data. In one embodiment, the user's cached data is stored in the cached database system 574 and the database system 522 uses the cached database system 574 to perform the operations disclosed in FIG. 44.

Access Network Switching— FIGS. 5, 45, 46, and 47

Switching Access by a User

Access between users and service providers vary based on quality and security, which in turn determine the costs of the access. Users typically have a need to switch between types of access depending on the service offered or what stage in the service the user is in. For example, a user browses the amazon.com website for books using a standard Internet access. When purchasing the books, the user needs a more secure Internet access to ensure that the user's credit card number is not stolen by a hacker.

One prior solution for enhanced security is data Virtual Private Networks (VPN). Data VPN's are relatively static constructions which allow the extension of a user network to another location by extending the network over leased lines or shared Internet/Intranet facilities. However, VPN's require service anticipation, planning and expense beyond what the typical Internet and Intranet users possess. VPN's also do not provide dynamic switching between accesses. Another prior solution is Local Area Network emulation (LANE). LANE utilizes ATM transports to extend the reach of the user network. However, most Internet and Intranet users do not possess ATM equipment and expertise, and the backbone network is still IP.

FIGS. 5 and 45 disclose one embodiment for switching access by a user in an example of the invention. A user

selects a different access to a service provider and is switched to the access without the user's connection to their access provider being interrupted. FIG. 45 depicts a flowchart for switching access by a user in an example of the invention. In this embodiment, a user using standard Internet access selects a premiere Internet access during the setup of a service application. There are numerous variations in switching between access paths such as from premiere Internet access path to a lower quality Internet access path, but only one embodiment is shown for the sake of simplicity.

FIG. 45 begins in step 4500. In step 4502, a user through the network device 512 transmits a request using a standard Internet access path through the service network 530 for a service application residing in the network device 562. The service application in the network device 562 receives the request and transmits a query for the quality of service (QOS) Internet access desired by the requester to the network device 512 in step 4504. The service application in the network device 562 then receives and processes a request to switch access to check if the user selected a premium QOS Internet access in step 4516. If the user did not select a premium QOS Internet access, the user and service application exchange packets using the standard Internet access via the network device 512, the access server 524, the service network 530, the access server 554, and the network device 562 in step 4516. The service application in the network device 562 returns the control to the user in step 4514 to select another service application in step 4502.

If the user selects a premium QOS Internet access, the service application in the network device 562 generates and transmits an authentication and authorization instruction for the premium QOS Internet access to the database system 522 to check if the switch of access is allowed. The database system 522 receives and processes the authentication and authorization instruction. The database system 522 then generates and transmits a premium access instruction to establish premium Internet access between the access server 524 and the access server 554 via the service network 540. In one embodiment, the database system 522 transmits the premium access instruction to the service network 540 to establish premium Internet access between the access server 524 and the access server 554. The database system 522 then generates and transmits the premium access instruction to the access server 524 to connect the network device 512 to the service network 540 for the premium Internet access. The database system 522 also generates and transmits the premium access instruction to the access server 554 to connect the network device 562 to the service network 540 for the premium Internet access.

Once the premium Internet access is established, the user and service application exchange packets using the premium Internet access via the network device 512, the access server 524, the service network 540, the access server 554, and the network device 562 in step 4512. The database system 522 then generates and transmits a disconnect instruction to the access server 524, the service network 540, and the access server 554 to tear down the premium Internet access in step 4518. The database system 522 then generates and transmits usage and performance statistics for the service application in the network device 562 in step 4520 before returning to step 4514. In one embodiment, the database system 522 uses a cross connect system 583 to perform the operations disclosed in FIG. 45.

Switching Access by a Service Provider

FIGS. 5 and 46 disclose one embodiment for switching access by a service provider in an example of the invention. The service provider selects a different access to a user and

41

is switched to the access without the user or service provider's connection to their access provider being interrupted. FIG. 46 depicts a flowchart for switching access by a service provider in an example of the invention. In this embodiment, the service provider selects a toll free premiere Internet service for a user using standard Internet access. The service provider pays for the changes in the toll free premium Internet service instead of the user—similar to toll free phone calls.

FIG. 46 begins in step 4600. In step 4602, the service application in the network device 562 waits for a request for a service. The user through the network device 512 transmits a request using a standard Internet access through the service network 530 for the service application in the network device 562 in step 4604. The service application in the network device 562 receives and processes the request. The service application then selects a toll free premiere quality of service (QOS) Internet access to the network device 512 in step 4606. The service application in the network device 562 generates and transmits an authentication instruction to the database system 522 in step 4608. The database system 522 receives and processes the authentication instruction. In step 4612, the database system 522 then generates and transmits a premium access instruction to establish the toll free premium Internet access between the access server 524 and the access server 554 via the service network 540.

Once the toll free premium Internet access is established, the user and service application exchange packets using the premium Internet access via the network device 512, the access server 524, the service network 540, the access server 554, and the network device 562 in step 4612. Once the service is completed, the database system 522 then generates and transmits a disconnect instruction to the access server 524, the service network 540, and the access server 554 to tear down the toll free premium Internet access in step 4614. The database system 522 then generates and transmits usage and performance statistics for the service application in the network device 562 in step 4616. Once the service is completed, the service application in the network device 562 returns the control to the user in step 4618 to select another service application in step 4602. In one embodiment, the database system 522 uses the cross connection system 583 to perform the operations disclosed in FIG. 46. In another embodiment, a third party such as the government performs the switching for surveillance or monitoring purposes.

Dynamic Switching Access

FIGS. 5 and 47 disclose one embodiment for dynamic switching access in an example of the invention. In a prior solution, a single access link from the user network to the access server is linked to a dedicated network in a single access session. However, users need to switch between different networks such as Internet and Intranets without tearing down the existing access session. No other systems allow the access server to be controlled by the data stream the access server is passing. Analog modems monitor the character stream for an escape sequence. The escape sequence notifies the modem that the data is for modem control and not for application data. However, no prior systems have implemented this functionality for a packet data network.

FIG. 47 depicts a flowchart for dynamic switching access in an example of the invention. In this embodiment, a user during an access session may switch networks without having the access session torn down. The access server 524 receives a control instruction from the user to switch from one dedicated service network 530 to another dedicated service network 540. FIG. 47 begins in step 4700. In step

42

4702, the access server 524 waits for a packet. In this embodiment, a request is in the format of a packet. The access server 524 then receives and processes the packet from the network device 512 in step 4704. The access server 524 checks if the packet is encoded for access control in step 4706. If the packet is not encoded for access control, the access server 524 processes the packet using standard access logic in step 4714 before returning to step 4702.

If the packet is encoded for access control, the access server 524 identifies the physical access path characteristics in step 4708. The access server 524 then checks if the access control is allowed in step 4710. If access control is not allowed, the access server 524 registers and logs an illegal access event in step 4716. The access server 524 then discards the packet in step 4718 before returning to step 4702.

If access control is allowed, the access server 524 generates and transmits an access control instruction to the database system 522 in step 4712. The database system 522 then receives and processes the access control instruction. The database system 522 identifies, authenticates, and authorizes the user and the requesting access server using the packet and path in step 4720. The database system 522 then retrieves the user access profile and the network device profile in step 4722. The database system 522 then checks if access control is allowed for the user and the network device based on the user access profile and the network device profile in step 4724. If access is not allowed, the database system 522 proceeds to step 4716.

If access is allowed, the database system 522 generates and transmits an access instruction to the access server 524 to update the access logic to switch to the service network 540 in step 4726. The database system 522 logs the access change in step 4727. In one embodiment, the database system 522 logs the access change in the audit system 572. The database system 522 also generates and transmits a reply with the new access status to the network device 512 in step 4728 before returning to step 4702. In one embodiment, the database system 522 uses the cross connect system 583 to perform the operations disclosed in FIG. 47.

Access Network Destination Control—FIGS. 5, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, and 59

Network Failover

Network devices become unavailable for a various reasons such as busy, failure, or overload. For network failover, when a destination network device is unavailable, network device requests are manually re-routed to a secondary network device. In order to avoid this manual rerouting, one prior art solution is the sharing of a network address between multiple processors in a box or multiple machines in a cluster. However, the requirement of destination and secondary devices residing in the same box or cluster makes this solution unacceptable for large systems.

Another prior art system is Network Address Translation, which translates a private network address to a public network address for a period of time. This system does not provide the capability of a network address to represent multiple failover destinations. Another solution is the Common Object Request Broker Architecture (CORBA), which provides the user a list of potential "logical" network devices to post the network device request. However, the user must implement CORBA interfaces and interactively select back up processes. The transaction context is lost when the primary fails. The user must re-establish the context. Unfortunately, none of these solutions provide an automatic translation from an unavailable destination network device to a secondary network device without operator

or user intervention, where the destination and secondary devices are not within the same box or cluster.

FIGS. 5, 48 and 49 disclose one embodiment for handling network address failover in an example of the invention. In a failover scenario, a destination network device fails or overloads, which makes the destination network device unavailable, and a secondary network device replicates the destination network device to give the user a fault tolerant appearance of the destination network device. Requests for the destination network device are translated to the secondary network device.

FIG. 48 depicts a flowchart for the access server 554 for network address failover in an example of the invention. FIG. 48 begins in step 4800. In step 4802, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 4804. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 4806, the access server 554 detects that the destination network device 562 fails and checks if the destination network address is enabled for failover in step 4806. If the destination network address is not enabled for failover, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 4812 before returning to step 4802.

If the destination network address is enabled for failover, the access server 554 checks if an address translation is active for the destination network device 562 in step 4808. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 4812 before returning to step 4802. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for failover in step 4810. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 4812 before returning to step 4802.

FIG. 49 depicts a flowchart for the database system 522 for network address failover in an example of the invention. FIG. 49 begins in step 4900. In step 4902, the database system 522 waits for a state event message or a timer event from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or a network device timer expires in step 4904. The database system 522 checks if the network device timer expired in step 4906.

If the network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout in step 4908. The database system 522 then retrieves and updates and the secondary network device's 564 profile to indicate that the destination network device 562 is unavailable in step 4910. In step 4912, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer has not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 4914. In step 4916, the database system 522 checks if the destination network device 562 is known. If the destination network device 562 is not known, the database system 522 generates

and transmits an unknown resource event before returning to step 4902. If the destination network device 562 is known, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics in step 4918. The database system 522 also resets the network device timer. The database system 522 then checks if the state event message is a failure or start event in step 4920.

If the state event message is not a failure or start event, the database system 522 checks if the load event threshold for the destination network device 562 is reached in step 4930. If the load event threshold is not reached, the database system 522 returns to step 4902. If the load event threshold is reached, the database system 522 proceeds to step 4926.

If the state event message is a failure or start event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 4922. In step 4924, the database system 522 checks if the state event message is a failure event. If the state event message is not a failure event, the database system 522 generates new active device profiles and then returns to step 4902. If the state event message is a failure event, the database system 522 proceeds to step 4926.

In step 4926, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address in step 4932 before returning to step 4902. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event in step 4928 before returning to step 4902. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 49.

Network Busy Forwarding

In a Public Switched Telephone Network (PSTN), a busy signal is sent to the caller when the called number destination is busy and thus unavailable. The caller can then select alternate actions such as delaying, retrying, or re-routing the request. Unfortunately, the PSTN handling of busy calls has not been applied to network devices in a data network. FIGS. 5, 50 and 51 disclose one embodiment for network busy forwarding in an example of the invention. When a destination network device 562 is busy, an access server 554 forwards the packets to a secondary network device 564. Requests for the destination network device are translated to the secondary network device.

FIG. 50 depicts a flowchart for the access server 554 for network busy forwarding in an example of the invention. FIG. 50 begins in step 5000. In step 5002, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5004. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5006, the access server 554 checks if the destination network device 562 is busy. If the destination network device 562 is not busy, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5012 before returning to step 5002.

If the destination network device 562 is busy, the access server 554 checks if an address translation is active for the destination network device 562 in step 5008. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step

5012 before returning to step 5002. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for failover in step 5010. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 5012 before returning to step 5002.

FIG. 51 depicts a flowchart for the database system 522 for network busy forwarding in an example of the invention. FIG. 51 begins in step 5100. In step 5102, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or the network device timer expired in step 5104. The database system 522 checks if a network device timer expired in step 5106.

If a network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout such as the network device being busy or unavailable in step 5108. The database system 522 then retrieves and updates and the secondary network device's 564 profile to indicate that the destination network device 562 is busy or unavailable in step 5110. The database system 522 also resets the network device timer. In step 5112, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer is not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 5114. In step 5116, the database system 522 checks if the destination network device 562 is known. If the destination network device 562 is not known, the database system 522 generates and transmits an unknown resource event before returning to step 5102. If the destination network device 562 is known, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics in step 5118. The database system 522 then checks if the state event message is a failure/busy or start/available event in step 5120.

If the state event message is not a failure/busy or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5130. If the load event threshold is not reached, the database system 522 returns to step 5102. If the load event threshold is reached, the database system 522 proceeds to step 5126.

If the state event message is a failure/busy or start/available event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 5122. In step 5124, the database system 522 checks if the state event message is a failure/busy event. If the state event message is not a failure/busy event, the database system 522 generates an address translation clear request in step 5132 before returning to step 5102. The address translation clear request adds a new active device profile or clears the existing translation. If the state event message is a failure/busy event, the database system 522 proceeds to step 5126.

In step 5126, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address if a busy

translation exists in the destination network device's 562 profile in step 5134 before returning to step 5102. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5128 before returning to step 5102. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 51.

Busy Flag

FIGS. 5 and 52 disclose one embodiment for a busy flag when the destination network device is busy in an example of the invention. When a destination network device 562 is busy and thus unavailable, an access server 554 replies to the user with a busy flag. FIG. 52 depicts a flowchart for the access server 554 for a busy flag when the destination network device is busy in an example of the invention. In one embodiment, the busy flag is a busy screen in Hyper-Text Markup Language. FIG. 52 begins in step 5200. In step 5204, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5204. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5206, the access server 554 checks if the destination network device 562 is busy. If the destination network device 562 is not busy, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5210 before returning to step 5202. If the destination network device 562 is busy, the access server 554 replies to the user with a busy flag in step 5208 before returning to step 5202.

Timeout

FIGS. 5 and 53 disclose one embodiment for forwarding if the destination network device times out in an example of the invention. In this embodiment, a destination network device 562 receives a request packet from a user but fails to reply to the user within a pre-determined time. The failure to reply within a pre-determined time indicates the destination network device 562 is unavailable. The access server 554 then redirects the packet to a secondary network device 564 to handle forwarding when the destination network device fails to respond with the pre-determined time.

FIG. 53 depicts a flowchart for the access server 554 for forwarding if the destination network device times out in an example of the invention. FIG. 53 begins in step 5300. In step 5302, the access server 554 waits for the next packet or event. The access server 554 then checks whether a reply timeout event is received in step 5304. If a reply timeout event is not received, the access server 554 then receives and processes the packet in step 5306. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5308, the access server 554 checks if an address translation is available for the destination network device 562. If an address translation is not available for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5318 before returning to step 5302. If an address translation is available for the destination network device 562, the access server 554 sets a reply timer and caches the packet for the secondary network device 564. The access server 554 then transmits the packet with standard access logic to the destination network device 564 in step 5318 before returning to step 5302.

In step 5312, the access server 554 receives the reply timeout event and retrieves the cached request. The access server 554 then processes the reply timeout event and the cached request in step 5314. In step 5315, the access server

554 then replaces the packet's destination network address with the cached secondary network address. The access server 554 then transmits the packet to the secondary network device 564 with the standard access logic in step 5318 before returning to step 5302. In one embodiment, upon reply, the access server 554 deletes the cached packet copy.

Scheduled Alias Resolution

Network devices become unavailable for various reasons such as busy, failure, or overload. Also, service provider need to utilize network resources to improve efficiency and avoid network device latency. In voice telephony signaling networks, an SS7 SCP or an automatic call distributor (ACD) distributes calls to end user call centers. The SCP receives a 800 TCAP query and performs a database lookup to translate the 800 telephone number into a destination telephone number. The SCP may use the requester information and the time of day to perform the database lookup. Some problems with this system are SS7 signaling must be used and the system is limited by voice network constraints. Also, the phone numbers only allow numeric numbers for aliases. In data networks, prior art known as Object Request Brokers (ORB) in a Common Object Request Broker Architecture (CORBA) advertise process resources to a requester. The ORB also registers multiple instances of the same type of process resource.

FIGS. 5, 54, and 55 disclose one embodiment for scheduled alias resolution in an example of the invention. In this embodiment, an alias, domain name/local name, network address, or macro for a network resource is translated to another alias for another network resource based on a configurable schedule. Advantageously, load balancing of network devices is improved by scheduling different alias resolutions for different times/conditions. One problem with CORBA is the requester and resources must implement CORBA interfaces. Scheduled alias resolution is compatible with existing data network DNS implementations. Also, the load balancing, security, and failover may be based on the user access profile and the network device state.

FIG. 54 depicts a flowchart for the access server 554 for schedule alias resolution in an example of the invention. FIG. 54 begins in step 5400. In step 5402, the access server 554 waits for the next request. The access server 554 then receives and processes a request from the network device 512 in step 5404. In this embodiment, the information including an alias is in the form of a request. In step 5406, the access server 554 checks if a current translation for the alias exists. If a current translation for the alias exists, the access server 554 translates the alias to the appropriate network address for a network device in user network 560 in step 5408. The access server 554 then replies to the network device 512 in step 5410 before returning to step 5402.

If a current translation for the alias does not exist, the access server 554 checks if a known access profile exists in the database system 522 exists for this user in step 5412. If a known access profile does not exist in the database system 522 for the user, the access server 554 generates and transmits an alias resolution failure message to the network device 512 in step 5414 before returning to step 5402. If a known access profile does exist in the database system 522 for the user, the access server 554 sets an external request timer in step 5416. The access server 554 then generates and transmits an alias translation request to the database system 522 in step 5418. The access server 554 then checks if a reply is received or the external request timer is expired in step 5420. If the reply is not received or the external request timer is not expired, the access server 554 returns to step

5420. If the reply is received or the external request timer is expired, the access server 554 checks if the reply was valid in step 5422. If the reply is invalid, the access server 554 returns to step 5414. If the reply is valid, the access server 554 then caches the alias translation for this user with time to live parameters in step 5424 before returning to step 5408.

FIG. 55 depicts a flow chart for the database system 522 for scheduled alias resolution in an example of the invention. FIG. 55 begins in step 5500. In step 5502, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 554 in step 5504. In step 5506, the database system 522 then checks if the access server requester is known in step 5506. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 5508. The database system 522 appends the reply to the packet and transmits the reply to the access server 554 in step 5510. The database system 522 replies with a decline message to the access server 554 in step 5512 before returning to step 5502.

If the access server requester is known, the database system 522 then checks whether the request is a translation appeal in step 5514. If the request is not a translation appeal, the database system 522 checks if the request is a security event in step 5516. If the request is a security event, the database system 522 increments a path/device security record in step 5522 before returning to step 5512. If the request is not a security event, the database system 522 registers an unknown event in step 5530 before returning to step 5502. If the request is a translation appeal, the database system 522 identifies and authenticates the user and the network device 512 in step 5518. If the access is not valid, the database system 522 returns to step 5522. If the access is valid, the database system 522 retrieves the user and network device's 512 profiles in step 5524. The database system 522 then generates the translation logic and calculates the translation time to live in step 5526. The database system 522 then appends the translation logic and the translation time to a reply to the access server 554. The database system 522 then replies with an approve message to the access server 554 in step 5528.

Forwarding

In a PSTN, a person may forward calls from their original phone number to another phone number where the person may be reached. Unfortunately, the PSTN handling of call forwarding has not been applied to network devices in a data network. FIGS. 5, 56 and 57 disclose one embodiment for destination controlled forwarding in an example of the invention. An access server 554 forwards the packets for a destination network device 562 that is unavailable to a secondary network device 564. Requests for the destination network device 562 are translated to the secondary network device 564.

FIG. 56 depicts a flowchart for the access server 554 for destination controlled forwarding in an example of the invention. FIG. 56 begins in step 5600. In step 5602, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5604. In this embodiment, the access server 554 receives and processes information in the form of a packet. The access server 554 then checks if the packet is a reply in step 5606. If the packet is a reply, the access server 554 retrieves and deletes the cached destination network address in step 5608 before proceeding to step 5614.

If the packet is not a reply, the access server 554 checks if an address translation is active for the destination network

device 562 in step 5610. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5614 before returning to step 5602. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for destination controlled forwarding in step 5612. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 5614 before returning to step 5602.

FIG. 57 depicts a flowchart for the database system 522 for destination controlled forwarding in an example of the invention. FIG. 57 begins in step 5700. In step 5702, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or a network device timer expires in step 5704.

The database system 522 processes the state event message and authenticates the destination network device 562 in step 5706. In step 5708, the database system 522 checks if the access server requester is known. If the access server requester is not known, the database system 522 generates and transmits an unknown requester event before returning to step 5702. If the access server requester is known, the database system 522 retrieves and updates the access server's 554 profile with availability and performance statistics in step 5710. The database system 522 then checks if the state event message is a failure/forward or start/available event in step 5712.

If the state event message is not a failure/forward or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5716. If the load event threshold is not reached, the database system 522 returns to step 5702. If the load event threshold is reached, the database system 522 proceeds to step 5722.

If the state event message is a failure/forward or start/available event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 5718. In step 5720, the database system 522 checks if the state event message is a failure/forward event. If the state event message is not a failure/forward event, the database system 522 generates and transmits an address translation clear request message to the access server 554 in step 5726 before returning to step 5702. If the state event is a start event, then a new availability profile is generated. If the state event message is a failure/busy event, the database system 522 proceeds to step 5722.

In step 5722, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address if a busy translation exists in the destination network device's 562 profile in step 5728 before returning to step 5702. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5724 before returning to step 5702. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 57.

Network Endpoint Availability Management

FIGS. 5, 58, and 59 disclose one embodiment for network endpoint availability management in an example of the invention. The features of failover, busy, busy flag, timeout, and forwarding are combined into this embodiment. FIG. 58 depicts a flowchart for the access server 554 network endpoint availability management in an example of the invention. FIG. 58 begins in step 5800. In step 5802, the access server 554 waits for the next packet or event. The access server 554 then checks whether a reply timeout event is received in step 5804.

If a reply timeout event is not received, the access server 554 then receives a data packet and processes the packet in step 5806. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5808, the access server 554 checks if the packet is a reply. If the packet is a reply, the access server 554 clears the cached request and clears a reply timer in step 5810 before returning to step 5804. If the packet is not a reply, the access server 554 checks if an address translation is active for the destination network device 562 in step 5812. If an address translation is active for the destination network device 562, the access server 554 proceeds to step 5824. If an address translation is not active for the destination network device 562, the access server 554 checks if the destination address is busy. If the destination address is not busy, the access server 554 proceeds to step 5830. If the destination address is not busy, the access server 554 checks if a forward address translation is available in step 5814. If the forward address translation is not available, replies to the user with a busy flag in step 5818 before returning to step 5802. If the forward address translation is available, the access server 554 proceeds to step 5824.

If a reply timeout event is received, the access server 554 receives the reply timeout event and the cached request in step 5820. The access server 554 then processes the reply timeout event and the cached request in step 5822. In step 5824, the access server 554 then replaces the packet's destination network address with the cached secondary network address. The access server 554 then transmits the packet to the secondary network device 564 with the standard access logic in step 5824.

In step 5826, the access server 554 checks if the forward address is busy. If the forward address is busy, the access server 554 returns to step 5818. If the forward address is not busy, the access server 554 sets a reply timer and caches the request in step 5828. In step 5830, the access server 554 then transmits the packet with standard access logic before returning to step 5802.

FIG. 59 depicts a flowchart for the database system 522 for network endpoint availability management in an example of the invention. FIG. 59 begins in step 5900. In step 5902, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives the state event message from the destination network device 562 or the network device timer expired in step 5904. The database system 522 checks if the network device timer expired in step 5906.

If the network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout such as the network device being busy or unavailable in step 5908. The database system 522 then retrieves and updates the secondary network device's 564 profile to indicate that the destination network device 562 is busy or unavailable in step 5910. The database system 522 also resets the network device timer. In step 5912, the database system 522 generates and transmits an

51

address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer has not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 5914. In step 5916, the database system 522 checks if the access server requester is known. If the access server requester is not known, the database system 522 generates and transmits an unknown resource event before returning to step 5902. If the access server requester is known, the database system 522 retrieves and updates the access server's 554 profile with availability, activity logs, and performance statistics in step 5918. The database system 522 then checks if the state event message is a failure, busy, forward or start/available event in step 5920.

If the state event message is not a failure, busy, forward or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5930. If the load event threshold is not reached, the database system 522 returns to step 5102. If the load event threshold is reached, the database system 522 proceeds to step 5926.

If the state event message is a failure, busy, forward, or start/available event, the database system 522 retrieves and updates the destination network device 562 and secondary network device's 564 profile in step 5922. In step 5124, the database system 522 checks if the state event message is a failure, busy, or forward event. If the state event message is not a failure/busy event, the database system 522 generates and transmits an address translation clear request to the access server 554 if a busy translation exists in step 5934 before returning to step 5902. If the state event message is a failure, busy, or forward event, the database system 522 proceeds to step 5926.

In step 5926, the database system 522 checks if a secondary network device 564 is available for forwarding. If a secondary network device 564 is available, the database system 522 checks if the state event message is a failure or busy event in step 5936. If the state event message is a failure or busy event, the database system 522 generates and transmits a busy address translation request message to the access server 554 in step 5932 before returning to step 5902. If the state event message is not a failure or busy event, the database system 522 generates and transmits a forward address translation request message to the access server 554 to change packets' destination network address to the secondary network address in step 5938 before returning to step 5902. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5928 before returning to step 5902. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 59.

Access Network Audit Server—

FIGS. 5, 60, 61, 62, and 63

Service Capability Monitor

FIGS. 5, 60, 61, 62, and 63 disclose one embodiment for service capability monitor in an example of the invention. In a prior system, Java applet technology and Sun's JINI project delivers communication and state mechanisms from the function provider to the function requesters. Java applet technology allows a mechanism for device control without any coordinating control agent. JINI provides a registry which maintains capabilities like a CORBA ORB as a means for primitive control. One problem is that JINI focuses on

52

the distribution of mechanisms instead of the interworking of mechanisms in a standardized way. In this embodiment of service capability control, a network access provider monitors the validity and state of the application level request to a network service provider. The network access provider monitors performance of their network and the service provider's network service requiring an understanding of the other provider's service requested. Access providers such as Internet service providers could mitigate their liability and increase the service's performance by extending their service view to the user base. Specific service environment access points may be monitored and service access rights and restrictions may be enforced. This embodiment provides an interworking mechanism that provides a dynamic interface to a service based network. The monitoring interface must be dynamic because different service have different potential states, different legal and illegal state transitions, and different communication mechanisms.

FIG. 60 depicts a flowchart for a firewall 526 for service capability monitor in an example of the invention. FIG. 60 begins in step 6000. In step 6002, the firewall 526 waits for a packet. The firewall 526 then receives and processes a packet from the network device 512 in step 6004. In step 6006, the firewall 526 performs standard firewall functions. The firewall 526 then copies the packets and transmits the packet to the destination and the database system 522 in step 6008 before returning step 6002.

FIG. 61 depicts a service capability monitor software architecture for a service capability monitor in an example of the invention. A service capability monitor software architecture 6100 comprises a service broker tokenizer process B 6112, a service broker state map process C 6114, a network A controller process A 6116, a service A signal tokenizer process E 6122, a service A state map process F 6124, a service A controller process D 6126, a service B signal tokenizer process H 6132, a service B state map process F 6134, and a service B controller process D 6136. The network A controller process A 6116 includes a service broker session state objects 6118. The service A controller process D 6126 includes a service A session state objects 6128. The service B controller process D 6136 includes a service B session state object 6138.

The network A controller process A 6116 is connected to the service broker tokenizer process B 6112, the service broker state map process C 6114, the service A controller process D 6126, and the service B controller process D 6136. The service A controller process D 6126 is connected to the service A signal tokenizer process E 6122 and the service A state map process F 6124. The service B controller process D 6136 is connected to the service B signal tokenizer process H 6132 and the service B state map process F 6134.

FIG. 62 depicts a flowchart for the network logic for service capability monitor in an example of the invention. FIG. 62 begins in step 6200. In step 6202, the database system 522 waits for the packet. The database system 522 then receives and processes the packet from the firewall 526 in step 6204. The database system 522 then identifies and authorizes the requester in step 6206. The database system 522 checks if the requester is known in step 6208.

If the requester is known, the network A controller process 6116 writes the packet to process B—service broker signal tokenizer process 6112 in step 6210. The process B 6112 then replies with the function token and parameters to the process A 6116 in step 6212. The database system 522 then checks if a service session state object (SSO) 6118 exists for this session in step 6214. If the SSO 6118 does not exist, the database system 522 creates a SSO 6118 for this session and

53

initializes the current state in step 6216. If the SSO 6118 exists, the database system 522 then proceeds to step 6218. In step 6218, process A 6116 writes the state token and current state to process C—service broker state map process 6114 in step 6218. The process C 6114 then replies with a transition token and parameters to the process A 6116 in step 6220. The process A 6116 applies the transition token and the parameters to the SSO 6118 to update the current state and setting actions in step 6222.

In step 6224, the database system 522 checks if there is an action indicated by the SSO 6118. If there is not an action indicated by the SSO 6118, the database system 522 proceeds to step 6228. If there is an action indicated by the SSO 6118, the database system 522 performs the transition action specified by the SSO 6118 in step 6226. In step 6228, the database system 522 updates the session sliding usage accumulators. The database system 522 then checks if there is an action in the SSO 6118 for this usage in step 6230. If there is no action, the database system 522 returns to step 6202. If there is an action in the SSO 6118, the database system 522 performs the usage action specified in the SSO 6118 before returning to step 6202.

If the requester is not known, the database system 522 checks if there is a network A state object in step 6234. If there is a network state object, the database system 522 proceeds to step 6238. If there is no network state object, the database system 522 creates a network A state object and initializes the current state in step 6236. In step 6238, the database system 522 increments network A state object usage for this event. The database system 522 then checks if there is an action in the network A state object for this usage in step 6240. If there is no action in the network A state object for this usage, the database system 522 returns to step 6202. If there is an action in the network A state object for this usage, the database system 522 performs the usage action specified in the network A state object in step 6242 before returning to step 6242.

FIG. 63 depicts a flowchart for the service logic for service capability monitor in an example of the invention. FIG. 63 begins in step 6300. In step 6302, process D service A controller process 6126 waits for the packet. The process D 6126 then receives and processes the packet from the process A network A controller 6116 in step 6304. The process D 6126 then identifies and authorizes the requester in step 6306. The process D 6126 checks if the requester is known in step 6308.

If the requester is known, the process D 6126 writes the request to process E service A signal tokenizer process 6122 in step 6310. The process E 6122 then replies with the state token and parameters to the process D 6126 in step 6312. The database system 522 then checks if a service session state object (SSO) 6128 exists for this session in step 6314. If the SSO 6128 does not exist, the database system 522 creates a SSO 6128 for this session and initializes the current state in step 6316. If the SSO 6118 exists, the database system 522 then proceeds to step 6318. In step 6218, the process D 6126 writes the state token and current state to process F—service state map process 6124 in step 6318. The process F 6124 then replies with a transition token and parameters to the process D 6126 in step 6320. The process D 6126 applies the transition token and the parameters to the SSO 6128 to update the current state and setting actions in step 6322.

In step 6324, the database system 522 checks if there is an action indicated by the SSO 6128. If there is not an action indicated by the SSO 6128, the database system 522 proceeds to step 6328. If there is an action indicated by the SSO

54

6128, the database system 522 performs the transition action specified by the SSO 6128 in step 6326. In step 6328, the database system 522 updates the session sliding usage accumulators. The database system 522 then checks if there is an action in the SSO 6128 for this usage in step 6330. If there is no action, the database system 522 returns to step 6302. If there is an action in the SSO 6128, the database system 522 performs the usage action specified in the SSO 6128 before returning to step 6302.

If the requester is not known, the database system 522 checks if there is a service A state object in step 6334. If there is a service state object, the database system 522 proceeds to step 6338. If there is no service state object, the database system 522 creates a service A state object and initializes the current state in step 6336. In step 6338, the database system 522 increments service A state object usage for this event. The database system 522 then checks if there is an action in the service A state object for this usage in step 6340. If there is no action in the service A state object for this usage, the database system 522 returns to step 6302. If there is an action in the service A state object for this usage, the database system 522 performs the usage action specified in the service A state object in step 6342 before returning to step 6342.

EXAMPLE

The following is one example of the access communication system that integrates many of the features described in the various embodiments of the inventions above. In the example below, a subscriber called Bank has many employees. The Bank has access to the network architecture 500 through an access provider called AccProv1. In this example, a bank employee is retrieving a stock quote from the yahoo.com website and then switches to request a streaming video from the yahoo.com website. Yahoo's web-server is running in the user network 562. Yahoo has access to the network architecture 500 through an access provider called AccProv2.

A system administrator for AccProv1 configures and optionally starts an access execution environment for the subscriber Bank in the local database system 570. The system administrator also starts a subscriber proxy for the Bank that runs in the access execution environment for the Bank. The subscriber proxy includes transport handlers, configuration handlers, and security handlers. The AccProv1 system administrator starts access execution environments for the maximum number of concurrent users in the local database system 570. The system administrator also starts generic proxies for users in the local database system 570. These generic proxies become user proxies after the user access profile is retrieved for the user.

The system administrator for AccProv2 starts an access execution environment for a service for Yahoo in the local database system 590. The system administrator also starts a service proxy for Yahoo's service that runs in the access execution environment for Yahoo. A system administrator for the Bank then sets up the user access profiles for each employee by inheriting attributes for the user access profile from capability classes or groups that the user belongs to. The user access profile includes the network shell for the user.

The Bank employee then logs on to the access server 524 using a user ID. The Bank system administrator set up. Besides user ID, a user may login to the access server 524 using a prepaid account code, a bank card number, an access card account code, or a user ID including a delimiter for user

access mobility. Upon connection, the access server 524 creates a device session in an access execution environment for the user. Then in response to the user login, the database system 522 retrieves the user access profile for the Bank employee. For a prepaid account code, a bank card number, an access card account code, and a user ID with a delimiter, the database system 522 may retrieve the user access profile from a database system external to the local database system 570. Also, the database system 522 may retrieve the user access profile from a database system external to the local database system 570 for global authentication.

Once the user access profile is retrieved, the access server 524 creates a user session in the access execution environment for the user and binds the device session with the user session. The configuration handler of the user proxy includes the network shell retrieved with the user access profile. The access server 524 then generates and transmits a list of available services in a service based directory to the Bank employee based on the user access profile.

The Bank employee then selects a service to check a stock quote for IBM by transmitting a request for service to the access server 524. The request for service may be in the form of a selection from the service based directory or an alias for alias translation for the network shell using domain name scoping or inbound DNS lookup. The access server 524 processes and transfers the request for service to the user proxy for the Bank employee. The user proxy transfers the request to the service proxy for Yahoo. The service proxy processes and transfers the request for service to Yahoo's web server. The database system 522 translates the service request for Yahoo to a private fulfillment address for one of Yahoo's web servers. The access server 554 then determines if a private destination address is available. If the private destination address is unavailable, the access server 554 could perform many actions including forwarding the request to another Yahoo web server or replying with a busy flag. The access server 524 also binds the user session and the device session with the selected service session by exchanging reference ID's.

The Bank employee then requests another stock quote for Oracle. The access server 524 then determines this stock quote was already cached based on the user's predictable pattern of requesting quotes for this company or a group of software companies. These prior requests for stock quotes of software companies by the Bank employee were audited to derive the Bank employee's predictable pattern. Alternatively, the request for the stock quote for Oracle could have been setup by the user in a script of commands to cache the quote in advance.

The Bank employee then requests a streaming video from Yahoo. The access server 524 switches the access for a premiere access based on the request for streaming video from the Bank employee. Alternatively, Yahoo initiates a switch of access for toll-free Internet service to the Bank employee who is a preferential customer. In order to be able to run the streaming video, the database system 522 also inherits user profile information from a class for streaming video user and updates the user access profile with the user profile information. The user proxy transfers the request for streaming video to the subscriber proxy. The subscriber proxy transfers the request to the service proxy. The service proxy then transfers the request to the Yahoo for one of their streaming video servers. Yahoo's streaming video server then provides the streaming video to the Bank employee.

Conclusion

The access network operates as described in response to instructions that are stored on storage media. The instruc-

tions are retrieved and executed by a processor in the access network. Typically, the processor resides in a server or database. Some examples of instructions are software, program code, and firmware. Some examples of storage media are memory devices, tape, disks, integrated circuits, and servers. The instructions are operational when executed by the processor to direct the network to operate in accord with the invention. The term "processor" refers to a single processing device or a group of inter-operational processing devices. Some examples of processors are computers, integrated circuits, and logic circuitry. Those skilled in the art are familiar with instructions, processors, and storage media.

Those skilled in the art will appreciate variations of the above-described embodiments that fall within the scope of the invention. As a result, the invention is not limited to the specific examples and illustrations discussed above, but only by the following claims and their equivalents.

1 claim:

1. A method of operating an access system including an access server to provide access between a user system and a plurality of communication networks that provide services to a user, the method comprising:

receiving a user login into the access server;

processing the user login to determine if the user is allowed access to the access system based on a local database system;

providing access to the access system to the user in response to the determination that the user is allowed access based on the local database system;

generating an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system;

in the second database system, receiving and processing the authorization query to determine whether the user is allowed access;

in the second database system, generating and transmitting an authorization response to the local database system;

receiving and processing the authorization response indicating whether the user is allowed to use the access system from the second database system; and

providing access to the access system to the user in response to the authorization response that allows the user to use the access system.

2. The method of claim 1 further comprising generating and transmitting a login query.

3. The method of claim 1 wherein processing the user login to determine if the user is allowed access to the access system based on a local database system is based on a user access profile.

4. The method of claim 1 further comprising determining if the second database system exists.

5. The method of claim 1 further comprising disconnecting a user's network device from the access server in response to the determination that the user is not allowed to use the access system.

6. The method of claim 1 wherein the login reply includes an access card account code.

7. The method of claim 1 wherein the login reply includes foreign network account information and further comprising identifying the second external database based on the foreign network account information.

8. The method of claim 1 further comprising:

receiving a request for access into a service control point;

processing the request for access to determine the destination for the request;

generating and transmitting a reply to route the request to the access server;

9. The method of claim 1 further comprising logging contract and settlement information.

10. An access system for providing access between a user system and a plurality of communication networks that provide services to a user, the access system comprising:

an access server connected to the user system and the plurality of communication networks and configured to receive and transmit a user logon from the user system to a local database system;

the local database system connected to the access server and configured to receive the user logon, process the user logon to determine if the user is allowed access to the access system based on the local database system, provide access to the access system to the user in response to the determination that the user is allowed access based on the local database system, generate an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system, receive and process an authorization response indicating whether the user is allowed to use the access system from the second database system, and provide access to the access system to the user in response to the authorization response that allows the user to use the access system; and

the second database system connected to the local database system and configured to receive and process the authorization query to determine whether the user is allowed access and generate and transmit the authorization response for the local database system.

11. The access system of claim 10 wherein the access server is configured to generate and transmit a logon query.

12. The access system of claim 10 wherein the local database system is configured to process the user logon to determine if the user is allowed access to the access system based on a user access profile.

13. The access system of claim 10 wherein the local database system is configured to determine if the second database system exists.

14. The access system of claim 10 wherein the access server is configured to disconnect a user's network device in response to the determination that the user is not allowed to use the access system.

15. The access system of claim 10 wherein the logon reply includes an access card account code.

16. The access system of claim 10 wherein the logon reply includes foreign network account information and the local database system is configured to identify the second external database based on the foreign network account information.

17. The access system of claim 10 further comprising:

a service control point connected to the user system configured to receive a request for access, process the request for access to determine the destination for the request, and generate and transmit a reply to route the request to the access server.

18. The access system of claim 10 wherein the local database system is configured to log contract and settlement information.

19. A software product for providing access between a user system and a plurality of communication networks that provide services to a user, the software product comprising:

database software operational when executed by a processor to direct the processor to receive the user logon, process the user logon to determine if the user is allowed access to an access system based on a local database system, provide access to the access system to the user in response to the determination that the user is allowed access based on the local database system, generate an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system, receive and process an authorization response indicating whether the user is allowed to use the access system from the second database system, and provide access to the access system to the user in response to the authorization response that allows the user to use the access system; and

a software storage medium operational to store the database software.

20. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to process the user logon to determine if the user is allowed access to the access system based on a user access profile.

21. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to determine if the second database system exists.

22. The software product of claim 19 wherein the logon reply includes an access card account code.

23. The software product of claim 19 wherein the logon reply includes foreign network account information and the database software is operational when executed by the processor to direct the processor to identify the second external database based on the foreign network account information.

24. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to log contract and settlement information.

* * * * *



(10) **Patent No.:** US 6,442,573 B1
(45) **Date of Patent:** Aug. 27, 2002

- | | | | | |
|----|----------|---|---------|------------------|
| WO | 98/04088 | * | 1/1998 | |
| WO | 98/26548 | * | 6/1998 | H04L/29/06 |
| WO | 99/04342 | * | 1/1999 | G06F/13/00 |
| WO | 99/44339 | * | 9/1999 | H04L/12/28 |
| WO | 99/54663 | * | 10/1999 | G06F/12/00 |
| WO | 99/56447 | * | 11/1999 | H04M/1/00 |
| WO | 00/29960 | * | 5/2000 | G06F/13/00 |

Hagawara-Sys-Com, "LUKIS (TM) JPEG Image Viewer (product information)," web page downloaded from www.h-scus.com/prod03 No. n/a, pp. 1-2, Apr. 2001.*

(21) Appl. No.: 09/458,849
(22) Filed: Dec. 10, 1999

Primary Examiner—Heather R. Herndon
Assistant Examiner—Jeffrey A. Rossi
(74) Attorney, Agent, or Firm—The Hecker Law Group

- (51) Int. Cl. G06F 15/00
(52) U.S. Cl. 707/500.1; 709/218; 358/403;
345/719
(58) Field of Search 345/302, 327,
345/719; 709/218, 215; 713/2; 707/500.1;
358/403

A method and apparatus for distributing picture mail to a frame device community is described. The present invention comprises one or more interconnected frame devices. Each frame device has a display region (e.g., an LCD) surrounded with a border region modeled to resemble a traditional picture frame. The border region may be comprised of wood, plastic, or any other aesthetically pleasing compound. Each frame device is configured to connect to an interconnection fabric to periodically obtain image data from a centralized repository and then display that data according to criteria established by an authorized user. The data repository is populated with image data via the image collection process. In one or more embodiments of the invention, the user may specify filter criteria which establishes what network addresses (e.g. picture mail address) are authorized to populate the data repository. The filter criteria and other information such as the behavior characteristics of each frame device are established and/or managed via a picture box. The picture box resides on a server computer and may be obtained by the user upon demand.

U.S. PATENT DOCUMENTS

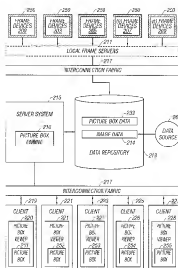
- | | | | | | |
|-----------|---|---|---------|---------------|---------|
| 4,742,345 | A | * | 5/1988 | Santo et al. | 345/107 |
| 5,394,166 | A | | 2/1995 | Shimada | 345/98 |
| 5,448,372 | A | * | 9/1995 | Axman et al. | 358/403 |
| 5,561,531 | A | * | 10/1996 | Funazaki | 358/403 |
| 5,612,741 | A | | 3/1997 | Loban et al. | 248/383 |
| 5,680,535 | A | * | 10/1997 | Harbin et al. | 345/473 |
| 5,706,457 | A | * | 1/1998 | Dwyer et al. | 345/835 |

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

- | | | | | | |
|----|----------|---|---------|-------|------------|
| WO | 92/05657 | * | 2/1992 | | H04N/1/21 |
| WO | 95/30300 | * | 11/1995 | | H04M/11/00 |
| WO | 95/31872 | * | 11/1995 | | H04N/7/10 |
| WO | 96/29639 | * | 9/1996 | | G06F/1/00 |
| WO | 97/47106 | * | 12/1997 | | H04L/9/00 |

17 Claims, 12 Drawing Sheets



U.S. PATENT DOCUMENTS

5,708,826 A	1/1998	Ikeda et al.	707/501.1
5,740,549 A *	4/1998	Reilly et al.	705/14
5,760,916 A *	6/1998	Dellert et al.	358/403
5,838,906 A	11/1998	Doyle et al.	709/202
5,862,297 A *	1/1999	Timmermans	358/403
5,864,387 A *	1/1999	Reed et al.	355/40
5,870,172 A	2/1999	Blume	708/105
5,898,600 A	4/1999	Isashi	708/105
5,905,492 A	5/1999	Straub et al.	345/744
5,913,088 A	6/1999	Moghadam et al.	396/311
5,923,736 A	7/1999	Shachar	379/93.17
5,923,738 A	7/1999	Cardillo, IV et al.	379/93.25
5,930,501 A *	7/1999	Neil	345/619
6,005,690 A *	12/1999	Suzuki et al.	358/403
6,025,827 A *	2/2000	Bullock et al.	707/500.1
6,037,989 A *	3/2000	Kim	348/441
6,058,428 A *	5/2000	Wang et al.	709/218
6,067,566 A *	5/2000	Moline	709/218
6,111,586 A *	8/2000	Ikeda et al.	345/619
6,167,469 A *	12/2000	Safai et al.	710/62
6,199,106 B1 *	3/2001	Shaw et al.	709/217

OTHER PUBLICATIONS

Kodak, Inc. Smart Picture Frame . . . for the Story Box (TM) Network (product description), downloaded from www.kodak.com/US/en/digital/assessories/smartFrame/ on Apr. 2001 (7 p. reprint).*

Heather Newman Free Press Staff, W., "Nifty Ways You Can Soup up Your Machine," Detroit Free Press No.—, pp. 3E, Aug. 1999.*

Anonymous, "Visions of the Future," Blaricum, Neatherlands: V+K Publishing No. ISBN 90 6611 591 2, pp. 12–13, Dec. 1996 (reprinted 1998).*

Takashi, D. "Doing Fieldwork In the High Tech Jungle", The Wall Street Journal, p. B1 & B22, Oct. 27, 1998.*

Kirschner, S. K., "Ideas that stick," Popular Science, vol. 254 No. 2, pp. 27, Feb. 1999.*

Okamoto et al., "Reproducing Device", translation of WO 99/54663, as provided by Ralph McElroy Translation Company (Washington DC), pp. 1–15+Figures.*

* cited by examiner

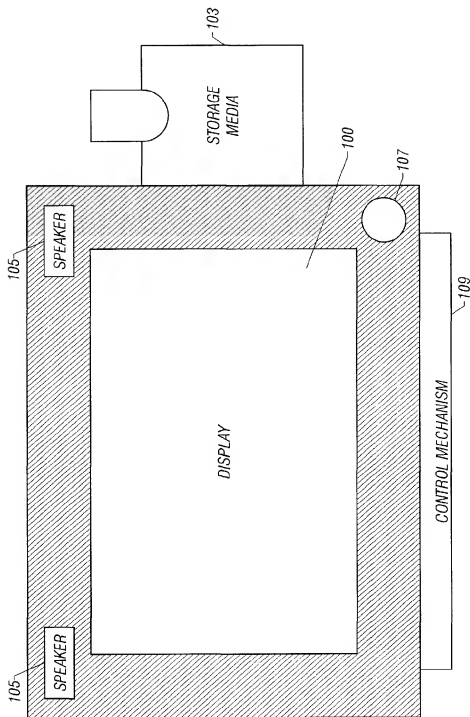


FIGURE 1A
(Prior Art)

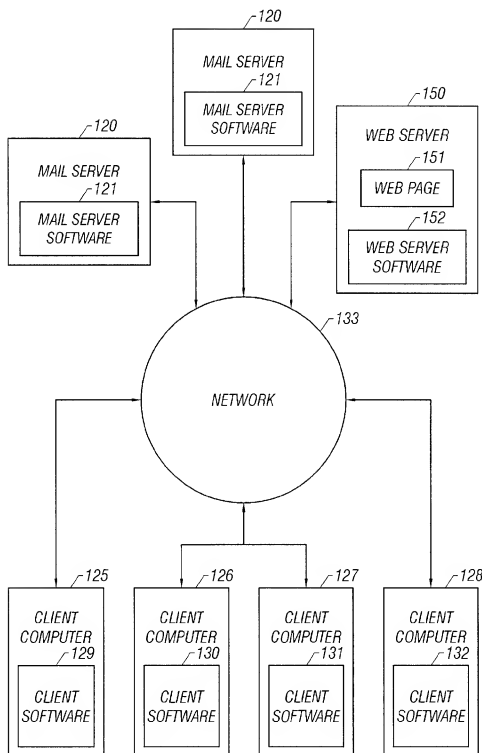


FIGURE 1B
(Prior Art)

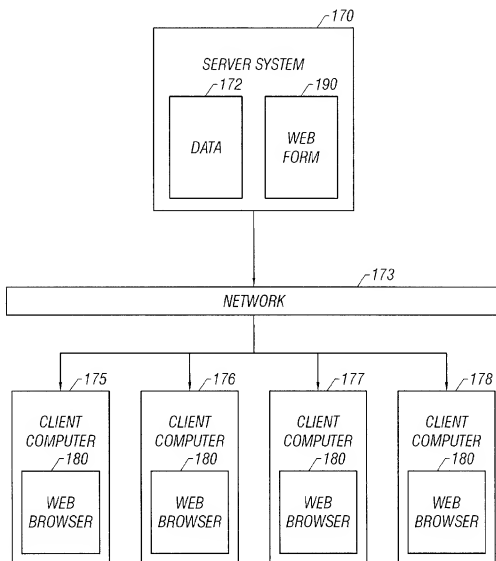


FIGURE 1C
(Prior Art)

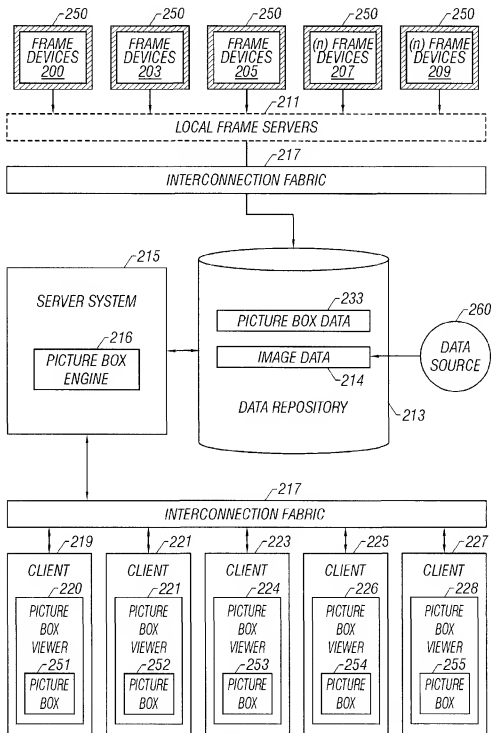


FIGURE 2A

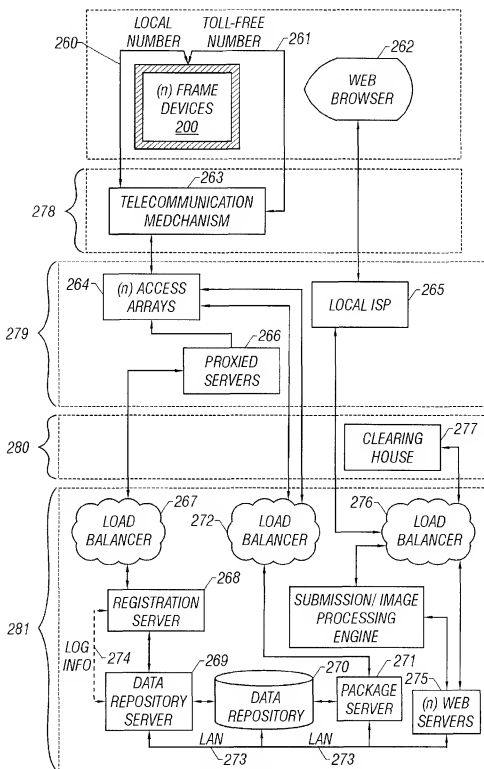


FIGURE 2B

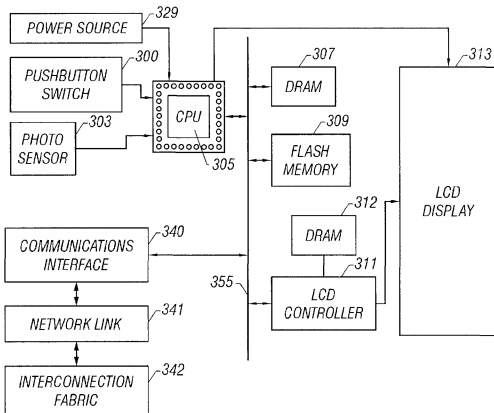


FIGURE 3

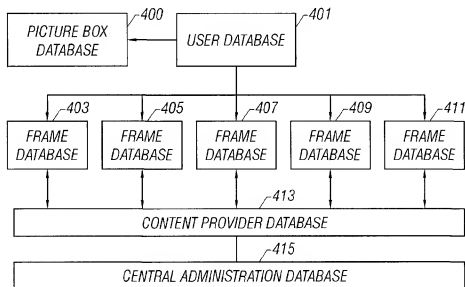


FIGURE 4

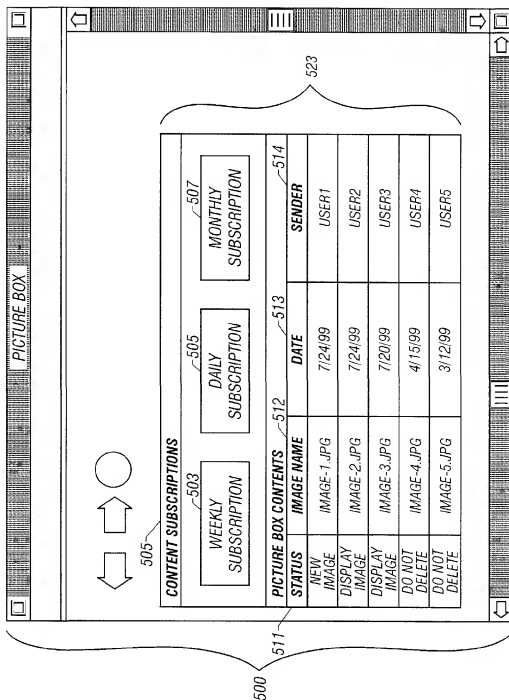


FIGURE 5

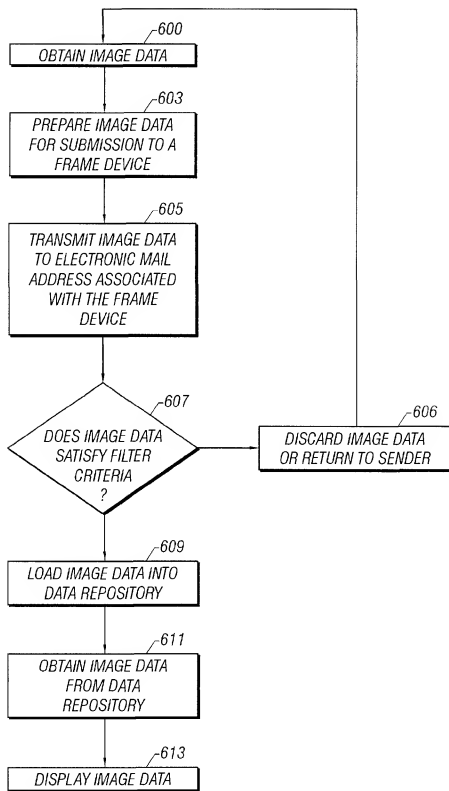


FIGURE 6

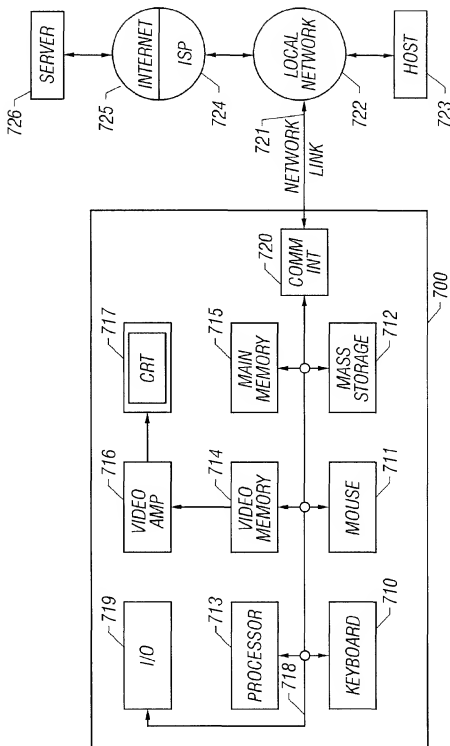


FIGURE 7

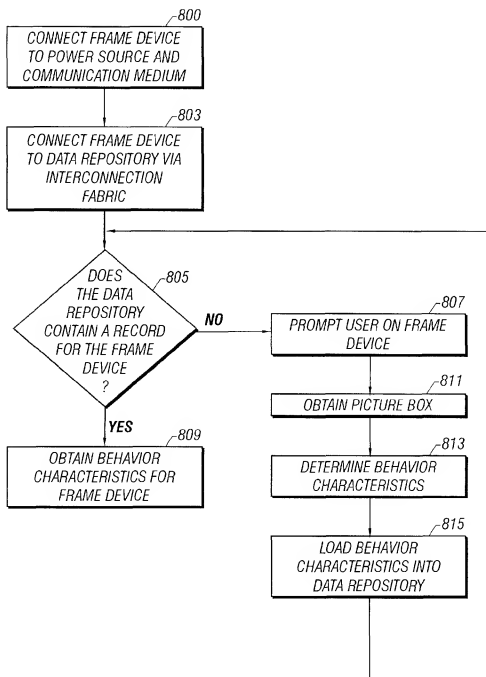


FIGURE 8

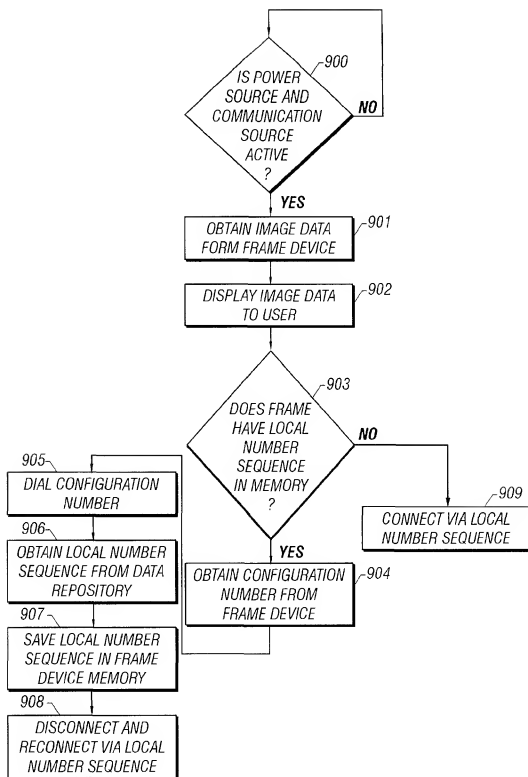


FIGURE 9

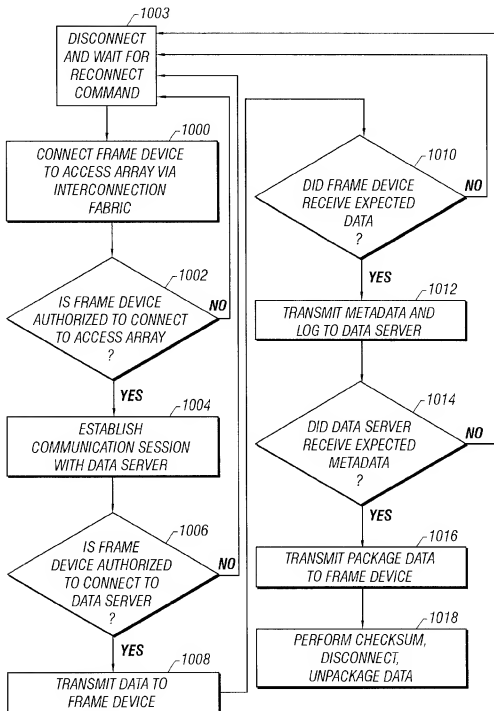


FIGURE 10

METHOD AND APPARATUS FOR DISTRIBUTING PICTURE MAIL TO A FRAME DEVICE COMMUNITY

FIELD OF THE INVENTION

This invention relates to the field of computer software. More specifically, the invention relates to a method and apparatus for distributing picture mail to a frame device community.

Portions of the disclosure of this patent document contain material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure as it appears in the Patent and Trademark Office file or records, but otherwise reserves all copyrights whatsoever.

BACKGROUND

Traditional Picture Frames

Picture frames provide a visually pleasing way to present photographs for display. People typically prefer to use picture frames, rather than digital display mechanisms (e.g. computer monitors), because of the aesthetic qualities associated with such frames. However, several problems and disadvantages result from the use of traditional picture frames.

For example, a problem that may be encountered by picture frame users is that changing the picture in the frame requires a series of manual steps. For example, a person who wishes to change the picture must manually remove the picture and replace it with a new one. Consequently, changing the picture requires that a person be physically located at the same place where the picture frame resides.

An additional problem with picture frames is that the frame does not have the ability to adjust the dimensions and/or size of the picture to fit within the display area. Pictures placed into the picture frame must correspond to the dimensions of the frame's display area. One prior art attempt to overcome these limitations is the use of various type of digital display device. The following section describes a prior art digital display device and discusses the inherent limitations of the device.

Digital Display Device

FIG. 1a illustrates a prior art device (e.g. the Sony PHD A55 CyberFrame™) capable of displaying digital images. However, there are several problems associated with this device. For example, the device lacks the ability to dynamically obtain image data from a networked data source and then display that data according to criteria established by an authorized user. The device shown in FIG. 1a consists of a display 100. Display 100 is an active matrix LCD screen configured to display digital video data and still image data. The data shown on display 100 is obtained from storage media 103. Storage media 103 is a memory medium capable of storing video and/or image data (e.g. a Memory Stick™).

The device contains software and/or hardware configured to playback MPEG or JPEG data files obtained from storage media 103. The device supports playback in a variety of different resolutions and can, for example, display JPEG data in UXGA (1600x1200), SXGA (1280x960), XGA (1024x768), or VGA (640x480). MPEG data files are shown on display 100 using a presentation mode that supports a resolution of 320x240 or a video mode that uses a resolution of 160x112.

The device also has built-in speaker(s) 105 and volume control 107. Speakers 105 provide a way to play back sound data introduced to the device via storage media 103. Volume control 107 allows the user of the device to adjust the decibel level. Control mechanism 109 contains the software and/or hardware utilized to control how data is displayed. For example, control mechanism 109 has a slide show mode that allows the user to display several images at varying intervals. Control mechanism 109 also provides the system with a way to delete unwanted images or keep certain images from being deleted. Control mechanism 109 is configured to provide other functionality, such as a sleep timer, brightness control, an index mode, an automatic angle detector, and a touchless sensor.

A problem with prior art mechanisms, such as the one illustrated in FIG. 1a, is that the user must physically provide storage media 103 to the device. Thus, a person who does not have physical access to the device cannot introduce new images into the device. Moreover, the device cannot be controlled from a remote location. For example, making changes to a web site cannot control the behavior of the control mechanism.

A further problem is that the functions offered by control mechanism 109 cannot be remotely updated, modified, or otherwise changed. For example, a new feature, such as the ability to obtain new images from a network, cannot be added by transmitting a software update to the device from a remote source. The device illustrated in FIG. 1a is isolated from other devices and therefore does not have the ability to communicate with such devices over a telecommunication medium.

Another limitation inherent in the design of the prior art device is that the device cannot automatically receive different types of image data. For example, the device cannot obtain different types of image data via an image delivery service. The user of the device illustrated in FIG. 1a cannot elect to periodically receive information relating to a particular topic such as the weather report. Computer networks are a prior art mechanism used to propagate data to devices connected to the network. The following section describes several techniques used to send and receive data across such computer networks and explains the limitations and disadvantages associated with such techniques.

Computer Networks

A network is an amalgamation of interconnected computers. Devices properly connected to the network may utilize the network to communicate with other devices coupled to the network. A server computer, for example, may use a network to transmit data to a client computer for display. There are several different techniques for propagating data to devices connected to the network. For example, electronic mail, client pull, and server push mechanisms all are examples of techniques that provide a way to transmit data to a client computer. However, these techniques lack a mechanism for establishing and then controlling the behavior of a device from a remote location. For example, these techniques cannot dynamically obtain image data from a networked data source and then display that data according to the behavior criteria established by an authorized user.

A. Electronic Mail Systems

The following section discusses electronic mail systems and points out the limitations associated with using such systems to distribute image data. An electronic mail system is an example of a prior art system used to distribute data to one or more recipients. The electronic mail system, for

3

example, provides users with a way to transmit data from one computer to another computer.

An example of an electronic mail system is shown in FIG. 1b. The system consists of one or more client computers 125-128 each having a client software program 129-132 configured to send and receive data, a network 133 configured to relay the data to one or more recipients, and a mail server 120 having server software 121 configured to store the data until it is retrieved by the designated recipient(s).

A user residing at client computer 125, for example, may use the system shown in FIG. 1b to transmit an image file to another user residing at a different computer (e.g., client, computer, 127). To send the file the user typically executes the appropriate client software program (e.g., client software 129), determines the data to be sent, and directs the program to send the data to a recipient. Data that is sent is routed across network 133 to the appropriate mail server. 120-123 associated with the recipient. Once mail server 120 receives the data, the server holds the data in the intended recipients account until requested by the recipient. For example, mail server 120 will store the data until the recipient residing at client computer, 127 executes a local version of client software program 129 and directs the program to obtain data from mail server 120.

A problem associated with using current electronic mail systems is that to receive and view images recipients must have physical access to a client computer containing a client software program configured to obtain mail data. For example, an electronic mail client (e.g., Microsoft Outlook Express) and/or a web browser (e.g., Microsoft Internet Explorer) must reside on client computer 127 in order for it to obtain data from server 120.

An additional problem with using current electronic mail systems is that in order to receive data the recipient must know how to navigate around the operating system and how to use the program utilized to obtain the data. For example, in some instances the recipient is required to manually configure the program utilized to obtain data. This requires that the recipient know the name and/or address of server 120 and be familiar with the settings required to login to server 120. Current electronic mail systems cannot, for example, automatically connect to an image source, obtain image data for display, and then automatically display the image data according to a set of predetermined preferences.

Instead, current systems require the recipient to manually perform a series of steps before the image data may be viewed. For example, receivers that use a Simple Mail Transfer Protocol (SMTP)/POP electronic mail system, are typically required to 1) open a client program configured to connect to the appropriate server, 2) direct the client to obtain the image data from the server, 3) select the electronic mail message containing the image data from a list of received messages, and 4) provide the image data to a client program configured to display the image data.

In some instances, however, the receiver cannot use the electronic mail client to view the image data, but instead must have an application specially configured to view the image data. If, for example, the receiver's electronic mail client cannot process images sent in the Tagged Image File Format (TIFF), the receiver must have an application capable of viewing TIFF files in order to view the image data transmitted by the sender. Thus, a problem with using current electronic mail clients to transmit image data is that such clients lack flexibility and require the user to manually open the electronic mail message to view the attached image data.

Another problem is that electronic mail client programs cannot obtain image data and then display a full sized view

4

of that data without requiring the user to manually intervene. Current electronic mail clients are not configured to periodically obtain and then automatically display images. Additionally, the receiver cannot control which images may be displayed and the frequency with which those images are displayed cannot be set based on the receiver's preferences. For example, mail clients lack the ability to automatically distribute data to a client computer according to a set of preferences determined by the user of the client computer. Furthermore, the receiver cannot set the behavior characteristics of the electronic mail client unless physically present at the client.

B. Client Pull:

The following section discusses the client pull technique and points out the limitations associated with using such systems to distribute image data. Client pull is an example of a technique used to download data from a server computer. When client pull is employed, data is requested by a client computer and then delivered by a server. For example, if web browser 131, executing at client computer 127, requests web page 151 stored on web server 150, it will cause web server 150 to execute web server software 152 which will in turn transmit web page 151 to client computer 127. The World Wide Web (WWW) is an example of a system that utilizes the client pull technique. The WWW is a segment of the Internet that utilizes an application layer protocol called the HyperText Transfer Protocol (HTTP) to disseminate and to obtain information from server computers (e.g., web server 150).

HTTP is a request/response protocol used with distributed, collaborative, hypermedia information systems. In operation, HTTP enables one computer to request data from another. For example, client 127 can use HTTP to communicate with web server 150 via network 133. In this scenario web server 150 acts as a data store for one or more web pages 151 and is capable of processing client 127's requests for such files. The web pages 151 stored on web server 150 may contain any type of data. For example, the files may contain data used to construct a form, image data, text data, or any other type of data. HTTP has communication methods that utilize the client pull technique to allow client 127 to request data from web server 150. Client 127 may use web browser 131 to initiate a request and thereby obtain web page 151.

Typically, web browser 131 requests at least one web page 151 from web server 150 and web server software 151 responds to the request by forwarding requested web page 151 to client 127. Once web page 151 arrives the connection is between client 127 and web server 150 is terminated. Client 127 uses web browser 131 to display requested web page 151. Web server 150 does not maintain any state information about the request once the connection is terminated. HTTP, which is frequently used to implement client pull, is, therefore, a stateless application protocol. That is, client 127 can send several requests to web server 150, but each individual request is treated independent of any other request. Web server 150 has no recollection of any previous request. Thus, for example, if a form is completed by the user and submitted to web server 150 for processing, the web server does not maintain a record of the data entered the form.

Once a file is sent from web server 150 to client 127 it becomes ready for display. Client 127's web browser 131 is typically used to format and display web page 151. Web browser 131 allows the user to request and view web page 131 (or any other web page) without having to learn a complicated command syntax. Examples of several widely

used web browsers include Netscape Navigator, Internet Explorer, and Opera. Some web browsers can display several different types of files. For example, web browser 131 may display files (e.g. web pages) written using the HyperText Markup Language (HTML), the JavaScript programming language, the ActiveX programming language, or the Portable Document Format (PDF). It is also possible to display various other types of files using language such as Standard Generalized Markup Language (SGML) or extensible Markup Language (XML).

A problem with utilizing client pull to distribute data, is that information about what data is to be disseminated to the client computer must be contained in the initial request. For example, if amuser residing at client 127 wishes to request web page 131, the user may do so by entering a Uniform Resource Locator (URL) and then transmitting the URL to web server 150. The URL is used to identify the name and location of web page 131 (e.g. web page 131 resides on web server 150). When web server 150 receives the request it transmits requested web page 131 to client 127. Web server 150 is not configured, for example, to transmit data that web server 150 determines client 127 needs (e.g. onboard software updates).

Another problem associated with client pull is that it does not provide a way to ensure data is displayed only to an intended recipient (e.g. a particular device and/or a particular user). Web server 150, for example, does not provide a way to ensure that data intended for display at a certain device is only sent to that device (e.g. a device with a unique serial number). An additional problem is that client 127 cannot be directed to request data without input or direction from a user residing at client 127. For example, a user residing at client 128 cannot control the behavior of client 127. Systems utilizing client pull do not have a mechanism for control the request patterns of multiple devices.

C. Server Push:

The following section discusses the server push techniques and points out the limitations associated with using such systems to distribute image data. Server push is a technique that provides a way to transmit a predetermined data set from a server computer to one or more client computers. Referring now to FIG. 1c, a system configured to perform one or more server push operations is shown. The system is generally composed of a server system 170, a network 173, and client computers 175-171. Each element of the system is configured to perform a task. Server system 170, for example, transmits data to any client computer 175-171 scheduled to receive data. Network 173 provides end-to-end connectivity for the system and thereby links server system 170 with client computers 175-171.

Each client computer 175-171 contains software configured to provide the user with an interface for operating the computer (e.g. an operating system). A web browser 180, residing at each client, provides users with a way to inform server 170 what type of data to send and how often data of that type should be sent. For example, if the user of client computer 175 wishes to have server 170 transmit stock market quotes (e.g. data 172) to client computer 175 on a daily basis, the user may direct server 170 to do so by obtaining a web form 190 from server 170 that provides such a option, filling the form out, and submitting the form to server 170. When server 120 receives the data provided by the user, it schedules the stock market quotes for dissemination at the interval specified by the user.

Server 120 may also be configured to disseminate electronic mail messages to client computers 125-131 on a regular basis. For example, if the user of client computer

129, or any other person authorized to control the behavior of server 120, directs server 120 to transmit an electronic mail messages to client computer 129 once a week, server 120 will distribute an electronic mail message to client computer 129 at the interval specified by the user.

Other types of data, such as document written using the HyperText Markup Language (HTML) may also be transmitted to client computers 175-171 for display. Such document may, for example, be embedded into the graphical user interface (e.g. the GCU desktop) or shown as screen savers. Examples, of such distribution systems include the active desktop included with Internet Explorer and the screen saver distribution scheme used by pointcast.

A problem with server push systems is that data is transmitted in a unidirectional manner. For example, data may be sent from server 170 to client computer 177 using server push techniques, but data cannot be transmitted from client computer 175 to client computer 177. Another problem associated with server push is that it does not provide a way to ensure data is displayed only to an intended recipient (e.g. a particular device and/or a particular user). Web server 170, for example, does not contain a mechanism for ensuring that data is only sent to devices having a certain serial number.

None of the prior art devices and/or systems described provide a way to distribute image data to a customizable frame devices. The prior art lacks a mechanism for remotely customizing the behavior of each frame device and does not have a distribution scheme configured to distribute image data to each frame device.

SUMMARY OF THE INVENTION

A method and apparatus for distributing picture mail to a frame device community is described. The present invention comprises one or more interconnected frame devices. A frame device is a self-configuring digital picture frame that obtains images for display from a repository of that may be accessed via an interconnection fabric (e.g. a computer network). If a person (referred to as a user) wants to display an image on the frame device the person may do so by transmitting the image data to the repository.

The frame device is programmed to connect to the repository and obtain new images for display. Once the frame device is connected to the network it may also use the opportunity to update the device's onboard software. In one or more embodiments of the invention, the frame device is configured to automatically execute the steps typically required for it to connect to the repository. Thus, the frame device is self-aware and requires only a minimal amount of input from the user. Once the frame device is connected to a power source and a telephone line, for example, the device is configured to automatically access the data repository without any further user input. The software that provides the frame device with its operational logic may be automatically upgraded without input from the user. For example, when the frame device connects to the data repository the device may elect to update and/or modify the operating system software located inside the frame device. Thus, if certain information necessary for normal day-to-day operation is lost (e.g. by a power outage), the frame device may recover without requiring input from the user.

The type of images and the frequency with which such images are displayed is configured remotely via a graphic user interface called a picture box. In one or more embodiments of the invention, the picture box is accessible via a web browser. The picture box provides a way to customize the behavior characteristics of the frame device to conform

7

to the wishes of the user. Each frame device has a display region (e.g., an LCD) surrounded by a border region modeled to resemble a traditional picture frame. The border region may be comprised of wood, plastic, or any other aesthetically pleasing compound. The border region may be, for example, an actual picture frame with a paper matte board that surrounds a thin LCD display region. Each frame device is configured to connect to an interconnection fabric to periodically obtain image data from a data repository and to then display that data according to criteria established by an authorized user.

The data repository is populated with image data via an image collection process. For example, in one or more embodiments of the invention, a networked data source (e.g., a client and/or server computer) populates the data repository with image data by submitting image data to a picture mail address associated with the target frame device. The user may specify filter criteria which establishes what network addresses (e.g. picture mail address) are authorized to populate the data repository. The filter criteria and other information such as the behavior characteristics of each frame device are established via a picture box.

When a user initializes a frame device, the user associated with the initialized frame device becomes a member of an online community. When the user initiates the frame device a picture box is created that allows the user to specify manage the behavior of the user's frame device. Each picture box is assigned a unique username (e.g., a picture address) and associated with one or more frame devices. The picture box provides the user with an interface for managing the behavior characteristics and filter criteria of one or more registered frame devices. For example, the picture box provides the user with a way to specify when the frame device should connect to the data repository to obtain an image and/or software update. When an update occurs new image data and/or information related to the operation and behavior of the frame device may be transmitted to the device.

In one or more embodiments of the invention, users may obtain a picture box by using a picture box viewer (e.g., a web browser). A picture box engine residing on a server system may generate the picture box. The picture box engine uses information stored in the data repository to generate the picture box that is displayed by the picture box viewer. For example, a user residing at a client computer may use a picture box viewer to request a picture box from a server computer. The server will respond to the request by executing the picture box engine, which in turn obtains the data used to generate a picture box from the data repository.

DESCRIPTION OF THE DRAWINGS

FIG. 1a illustrates a device configured to display digital images.

FIG. 1b illustrates the components used by an electronic mail system and/or a client pull system such as the World Wide Web.

FIG. 1c illustrates a system configured to push data from a server computer to one or more client computers.

FIG. 2a is a block diagram that illustrates the components utilized by one embodiment of the invention.

FIG. 2b illustrates the system components utilized by one or more embodiments of the invention.

FIG. 3 is a block diagram that illustrates the internal components of the frame device.

FIG. 4 illustrates what type of data may be stored in the data repository.

8

FIG. 5 is an illustration of a picture box viewer containing a picture box.

FIG. 6 is a flow chart illustrating the process utilized by one embodiment of the invention to prepare an image for display in a frame device.

FIG. 7 illustrates an embodiment of the invention that may be utilized to access the picture box interface.

FIG. 8 is a flow chart illustrating the process utilized by one or more embodiments of the invention to register a frame device.

FIG. 9 illustrates the process utilized by one or more embodiments of the invention to initialize a frame device.

FIG. 10 illustrates the process used by one or more embodiments of the invention to obtain new image data and/or software updates.

DETAILED DESCRIPTION

A method and apparatus for distributing picture mail to a frame device community is described. In the following description numerous specific details are set forth in order to provide a more thorough understanding of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In other instances, well-known features have not been described in detail so as not to obscure the invention.

General Overview

The present invention comprises one or more interconnected frame devices. A frame device is a self-configuring digital picture frame that obtains images for display and/or software from a data repository via an interconnection fabric (e.g., a computer network). Each frame device is configured to automatically interface with the data repository to obtain image data and to then display that data according to a set of preferences. For example, a frame device may dynamically obtain image data from a networked data source (e.g., a client computer), store that data, and then display that data according to criteria established by an authorized user.

Each frame device provides users with a simplified interface that minimizes the amount of complexity presented to the user without limiting the ability of the device to perform an advanced set of functions. For example, each frame device is programmed to automatically configure itself without obtaining input from the user. So long as the frame device is connected to a power source and a communication source it may remain operational without obtaining input from the user. The software that controls the behavior of each frame device may also be updated or modified without requiring additional input from the user. One embodiment of the invention provides a simplified pushbutton interface that provides users with a mechanism for manually directing the behavior of the device. However, the device will continue to operate even if users do not direct it to via the pushbutton interface. Each frame device is part of a community of interrelated components that operate in concert to provide image data to a family of one or more frame devices. The details of the system are further described below.

Image data is typically stored in a data repository (e.g., a database). Users or people authorized to submit images may transmit image data to the data repository which in turn provides the image data to an associated frame device by sending the image data via picture mail. A picture box, which comprises functionality further described below, provides users with a mechanism for controlling the behavior

characteristics of each frame device. Thus, users may control the picture box without touching the pushbutton interface on the device. When the picture box is created the user associated with the picture box becomes a member of a picture mail community. Members of the picture mail community are each provided with one or more picture mail addresses.

Each frame device contains onboard software designed to automatically connect to an image delivery service. The image delivery service provides a mechanism for transmitting certain types of images to a frame device based on a set of predefined criteria. The specific aspects of the image delivery service are further described below. When a frame device is connected to a network it may automatically execute an initialization process without obtaining any additional input from the user. In one embodiment of the invention this is referred to as the self-configuration protocol. The initialization process is further described below. The data repository may act as an image relay mechanism that processes any image data that is submitted to the system via an image collection/submission engine. The functionality provided by the image collection/submission engine is discussed further below. Once the image data is processed by the collection/submission engine the data repository stores the image data until the frame device connects to the repository for an update. Each frame device is designed to connect to a data source such as the data repository to obtain an update (e.g. image data or software data) using a conversation/security protocol.

Automatic Configuration Without User Input

The device is configured to automatically execute the steps required to connect to the repository and does not require input from the user to obtain new images and/or update the onboard software. In one or more embodiments of the invention, for example, the frame begins to acquire configuration information by obtaining the toll free phone number stored in the devices' memory and using that number to connect to a server computer. The server responds by analyzing the caller ID information contained in the connection signal (e.g. ANI) to determine what local phone number or phone numbers the frame device should utilize to inexpensively connect to the data repository. Based on the location of the frame device during dial-in, the server computer provides the frame device with information that directs the device when and where it should dial. The caller ID information, for example, provides the frame device with a way to discern what geographic region the device is located in and based on that information the device knows when to connect to the data repository for an image and/or software update. If the server determines the frame device is in located in Los Angeles, Calif. the server may inform the device to connect at 12:00 am Pacific Daylight Time (PDT) using a phone number located in the Los Angeles area. The local phone number information may be referred to as a localized number sequence.

The frame device responds by storing the information provided by the server in memory and then disconnecting from the toll free phone number without requiring any input from the user. Once the device is disconnected from toll free phone number the frame redials using one of the local phone numbers obtained while the device was connected to the toll free phone number. The local phone number provides the frame device with a mechanism for connecting to the data repository to obtain new images and/or software updates. The frame device typically uses the local phone number information to connect to the data repository at the time

specified by the server without requiring any input from the user of the device. Thus, the frame device is self-configuring and self-maintained. If certain information necessary for normal day-to-day operation is lost (e.g. by a power outage or by the user physically moving the frame device), the frame device may recover without requiring input from the user.

Self-Upgrading Without User Input

The software that provides the frame device with its operational logic may be automatically updated and/or upgraded without input from the user. For example, when the frame device connects to the data repository the device may elect to update and/or modify the operating system software located inside the frame device. The functionality of the pushbutton switch or behavior characteristics of the frame device, for example, may be altered by updating the onboard software that controls the device. Transmitting an update to the frame device may modify any characteristic that may be controlled via onboard software.

Simplified User Interface

Users are not required to perform any complex activities to configure the device. The device may, however, connect to the data repository when manually directed to by the user. The frame device contains a simplistic pushbutton interface that provides novice users with a mechanism for directing the behavior of the device. For example, one or more pushbuttons provides users with a way to manually initiate a connection to the data repository. Other functions such as dimming the LCD or cycling through a set of available images (e.g. slide show mode) may also be provided by the pushbutton interface. A detailed discussion of the pushbutton interface located on the frame device follows below.

The Components of a Frame Device

FIG. 3 shows a block diagram illustrating the components of a frame device as it is used by one embodiment of the invention. Each frame device comprises a Central Processing Unit (CPU 305), memory (e.g. flash memory and DRAM), and telecommunication hardware and/or software configured to provide the device with a mechanism for connecting to a data repository. Onboard software stored in memory provides each frame device with a set of behavior characteristics that may be customized by the user, or by any other person authorized to alter the frame device's behavior characteristics. CPU 305 is responsible for executing tasks for the frame device. In one or more embodiments of the invention, CPU 305 is a microprocessor manufactured by Cirrus Logic, such as the EP7211 ARM processor, or a microprocessor manufactured by Intel, such as the 80x86, or Pentium processor. However, any other suitable CPU, microprocessor, or microcomputer may be utilized.

CPU 305 communicates with the other components of the frame device utilizing system bus 335. Bus 335 may contain, for example, thirty-two address lines for addressing flash memory 309 or DRAM 307. The system bus 335 may also include, for example, a 64-bit or 32-bit data bus for transferring data between and among the components, such as CPU 305, flash memory 309, DRAM 312, and DRAM 307. Alternatively, multiplex data/address lines may be used instead of separate data and address lines.

Display 313 represents the display hardware used to render a visual representation of the image data. In one or more embodiments of the invention, display 313 is a Liquid Crystal Display (LCD). However, the invention also anti-

pates the use of other display mechanisms that are capable of rendering an image for display. Flat panel technologies such as plasma displays, Field Emission Displays (FED), or improved Cathode Ray Tube (CRT) monitors, for example, may also be utilized to display images. Display 313 is surrounded with a border region modeled to resemble a traditional picture frame. Display 313 is a low profile LCD designed to minimize the prevalence of display 313. Thus, the emphasis is on the picture frame rather than the LCD. Controller 311 controls the image data output to display 313. For example, if LCD hardware it utilized to display image data, controller 311 is a LCD controller. DRAM 312 is the memory controller 311 uses to prepare image data for display on display 313. DRAM 312, for example, may be utilized for purposes of frame buffering. Dithering may be performed on the frame device in order to improve the quality of images displayed on display 313.

Flash memory 309 provides the frame device with storage space for the image data. For example, image data held in the data repository may be copied into flash memory 309 when a connection to the data repository occurs. In one or more embodiments of the invention, the image data held in flash memory 309 is stored in a compressed format and decompressed when the data is output to display 313. Flash memory 309 may hold the onboard software that defines the functionality of the device. The mechanism and data used to control the behavior characteristics of the frame device is also stored in flash memory 309. For example, flash memory 309 may hold the local phone number utilized to connect to the data repository. In case the local phone number cannot be reached a toll-free phone number also resides in flash memory 309. The toll-free phone number may be used if the local number fails. Other information that is required to initiate a communication session with the data repository may also reside in flash memory 309.

Flash memory 309 may store timing information. For example, the time the frame device is scheduled to dim display 313, the time the frame device is scheduled to connect to the data repository for an update, and the current time are stored in flash memory. In one or more embodiments of the invention, relative time is utilized to indicate the current time. Relative time is kept by keeping track of the number of clock cycles performed by CPU 305. However, the invention contemplates the use of other mechanisms for keeping time. For example, the current time may be obtained from a central source located on the frame device. The current time may also be obtained by accessing other networked devices such as a server computer containing a system clock.

In one or more embodiments of the invention, flash memory 309 is configured to store the software release number and/or version number of the onboard software. When a connection to the data repository occurs the version of software held resident in flash memory 309 is compared to the version held in the repository. If the version of software held in flash memory 309 is older than the version in the data repository then a newer version is loaded into flash memory 309. Thus, the frame device is configured to automatically update its software without requiring input from the user. DRAM 307 provides the frame device with an execution space for general-purpose tasks (e.g. frame buffering). Modulator demodulator 319, for example, may utilize DRAM 309 as a place to process downloaded data.

The function of pushbutton switch 300 is to provide the user with an interface for controlling certain attributes of the frame device. If, for example, the user depresses pushbutton switch 300 the frame device will illuminate display 313 so

that images shown on the display become visible. Pushbutton switch 300 is also used as a frame advance. Thus, pushbutton switch 300 provides the user with a way to navigate through the images held in flash memory 309. If ten images currently reside in flash memory 309, the user may utilize pushbutton switch 300 to cycle through the ten images in flash memory 309 available for display. One embodiment of the present invention contemplates using pushbutton switch 300 to force the frame device to update its image library. For example, depressing pushbutton switch 300 for an extended duration of time (e.g. several seconds) causes the device to initiate a connection to the data repository. Once the connection is established the frame device utilizes the information contained in the data repository to supply the frame device With a new set of images. Pushbutton switch 300 also provides a way for the user to alter the backlight level of display 313. In one embodiment of the present invention, pushbutton switch 300 is configured to alter any of the attributes of the frame device provided via software. Thus, it is possible to periodically change the features controlled by pushbutton switch 300 to meet the demands of the user.

The invention also contemplates the use of multiple pushbutton switch 300's. If multiple pushbutton switch 300's are present each pushbutton switch 300 may be configured to perform different functions. For example, one pushbutton switch 300 may be utilized as a dimmer button that controls the brightness of the frame, whereas the second pushbutton switch may be configured to act as a main button that has multiple functions. In one or more embodiments of the invention, the pushbutton switch provides a mechanism for marking images for deletion. The pushbutton switch may also be utilized to identify images that are to be saved or archived. Photo sensor 303 supplies the frame device with a mechanism for automatically dimming display 313 when certain conditions, such as the passage of time or the activation of pushbutton switch 300, occur. The frame device is operational without photo sensor 303. Therefore, photo sensor 303 is not required, but may be included if dimming functionality is desired.

Communication interface 340 provides a two-way data communication coupling via a network link 341 to interconnection fabric 342. Communication interface 340 may be implemented in software or hardware form. In one or more embodiments of the invention, data communication interface provides a codec for optimizing data throughput to network link 341. Interconnection fabric 342 represents any type of network configured to transmit data. For example, interconnection fabric 342 may represent the Internet or any other type of easily accessible computer network.

If communication interface 340 is an integrated services digital network (ISDN) card or a modem, communication interface 340 provides a data communication connection to the corresponding type of telephone line, which comprises part of network link 341. Data is then transported across network link 341 to interconnection fabric 342. If communication interface 340 is a local area network (LAN) card, communication interface 340 utilizes a compatible LAN as network link 341 to transmit data to interconnection fabric 342. Wireless links are also possible. In any such implementation, communication interface 340 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information. These signals are transmitted across network link 341 to interconnection fabric 342. Therefore, interconnection fabric 342 couples the frame device to the data repository.

Power source 329 provides the frame device with the electrical current necessary to properly run the device. In one or more embodiments of the invention, power source 329 obtains an electrical signal from a standard wall plug and utilizes a transformer unit, a voltage converter and an econo reset device to prepare the signal for use by the frame device. Power source 329 may also obtain an electrical signal from a battery, solar power, or any other source that can generate the appropriate amount of current required to adequately power the frame device.

System of Interrelated Components

FIG. 2a is a block diagram that illustrates the components comprising one embodiment of the invention. In the embodiment illustrated, the system comprises one or more frame devices 200-209 configured to connect to data repository 213 via an interconnection fabric 217. Frame devices 200-209 are designed to resemble traditional picture frames. For example, each frame device contains a border region 250 that enhances the aesthetic qualities of the device. The border region 250 may be comprised of wood, plastic, metal, or any other compound that is pleasing to the eye. Each frame device 200-209 may have a border region 250 that appears to be different from the border region 250 of another frame device. For example, the border region 250 for frame device 200 may appear different than the border region 250 attached to frame device 209. In one or more embodiments of the invention, border region 250 is an actual picture frame sized to surround the display of frame device 209. Border region 250 may contain a matte board designed to complement the picture frame.

However, unlike a traditional picture frame, the images shown in the display region of frame devices 200-209 can be periodically modified. For example, frame device 200 can be configured to show a first image for a certain duration of time and a second image for another duration of time. Image data 214, for example, can be shown in the display region once it is obtained from the data repository.

Frame devices 200-209 connect to data repository 213 to obtain image data 214 via interconnection fabric 217. In one or more embodiments of the invention, frame devices 200-209 use one or more local frame servers 211 to connect to interconnection fabric 217. An Internet Service Provider (ISP), modem pool, or any other mechanism capable of connecting frame devices 200-209 to interconnection fabric 217 is an example of a local frame server. However, local frame servers 211 are not required to achieve connectivity to interconnection fabric 217. Frame devices 200-209, for example, may connect directly to interconnection fabric 217 and thereby bypass local frame servers 211.

In the invention, interconnection fabric 217 is any of multiple suitable communication paths for carrying data between frame devices 200-209 and data repository 213. Interconnection fabric 217 may be, for example, a local area network (LAN) implemented as an Ethernet network. Any other local network may also be utilized. The invention also contemplates the use of wide area networks such as the Internet and/or World Wide Web. Interconnection fabric 217 may be implemented using a physical medium such as a wire or fiber optic cable, or it may be implemented in a wireless environment. For example, the invention contemplates the use of dedicated, dial-up, or shared communication inter-

connections. The data repository is populated with image data 214 via an image collection process. Image data 214 is inserted into data repository 213 by data source 260. In one or more

embodiments of the invention, data source 260 populates data repository 213 with image data by obtaining the data from a storage medium such as a hard drive, preparing the data for submission, and transmitting the data to data repository 213. A program executing at data repository 213 determines whether the image data satisfies certain filter criteria. For example, the program may determine whether the image data was transmitted from a member of the users buddy-list and/or address book. If the image data does not satisfy the filter criteria it may be discarded or returned to the original data source. For example, in one or more embodiments of the invention, a networked data source (e.g. a client and/or server computer) populates the data repository with image data by submitting image data to a picture mail address associated with the targeted frame device. In one or more embodiments of the invention, the picture mail address is a network address (e.g. an IP address) that is associated with a frame device. The user may specify filter criteria (e.g. via a picture box) which establishes what addresses (e.g. picture mail address) are authorized to populate the data repository. Users who join the picture mail community by connecting one or more frame devices to the interconnection fabric may, for example, pass images data to other members of the community via a picture mail address. However, users are not required to belong to the picture mail community to submit images to a frame device that does belong to the picture mail community. In addition to storing the image data, and filter criteria, data repository 213 also comprises data that may be used to define the behavior of frame devices 200-209. The structure and arrangement of the data repository itself will be discussed further below. In one or more embodiments of the invention, the filter criteria (e.g. who is permitted to send images to a frame device) and other information held in data repository 213 such as the behavior characteristics of each frame device are established via a picture box.

Users may use the picture box to customize the behavior characteristics associated with each frame device 200-209. Specifying the time and type of image data 214 frame device 200 is to obtain is an example of a customizable behavior characteristic. However, any feature and/or function offered by the frame devices 200-209 may become a behavior characteristic.

Each picture box 229-233 may provide a way, for example, to specify how often a certain frame device retrieves new image data from the data repository. For example, the user of client computer 225 may use picture box 232 to specify how, often frame device 207 connects to data repository 213. Picture box 232 can also be configured to provide the user of client computer 225 with a mechanism for controlling what images are deliver to frame device 207. For example, the user could specify that images should only be accepted from people listed in the user's buddy-list, friendly-sender list, or address book. If the user of client computer 225 wants to subscribe to an image delivery service, this preference may also be expressed via picture box 232. The image delivery service regularly delivers images that illustrate weather reports, art collections, greeting cards, movie posters, post cards, live camera data, or any other type of customized data to the user's frame device. Each image delivery service may be separately subscribed to and the device may be unsubscribe at any time.

The user may also utilize the picture box to define filter criteria (e.g. to specify who is authorized to populate a certain frame device with new images). For example, the user of client computer 221 may use picture box 232 to indicate that only images received from certain picture mail

addresses are to be held for display in data repository 213 and forwarded to frame device 203 upon request. Each individual picture box 229-233 controls the behavior characteristics of one or more frame devices 200-209. Picture box 220, for example, may be configured to control frame device 203 and 209.

In one or more embodiments of the invention, the picture box engine 216 is responsible for obtaining the picture box data 233 needed to generate a picture box from data repository 213 and transmitting a visual representation of the picture box to the appropriate picture box viewer. For example, a user who has permission to control frame device 209 may control the behavior characteristics of the device via a picture box. The user may obtain picture box 251 by using picture box viewer 220 to request picture box 251 from server system 215. For example, if a user residing at client computer 219 uses picture box viewer 220 to request picture box 251 from server computer 215 the server responds by executing picture box engine 216, which in turn obtains picture box data 233 from data repository 213. In this instance, picture box data 233 is associated with frame device 209. However, the picture box may be associated with any frame device 200-209 the user is authorized to control. The user may control multiple frame devices via a single picture box. A single user can also elect to use multiple picture boxes to control multiple frame devices.

It is possible for the user to modify the onboard software embedded into each frame device 200-209 by making changes to the picture box. When a change is made it is reflected in data repository 213 and communicated to the frame device when the device connects to data repository 213 for an update. Other authorized parties, such as the picture box provider, can also modify and/or update the functionality provided by the onboard software by indicating to the frame device when it connects to the data repository that new software is available for it to obtain. When the frame device receives such a message it may automatically obtain and install the new software update. Thus, authorized users can automatically add new functionality and features into the frame device without prompting the user for input.

Each client computer 219-227 is configured to run a picture box viewer 220-228. For instance, client computer 219 is configured to execute picture box viewer 220. Picture box viewer 220 provides the user of client computer 219 with a visual representation of picture box 229. In one or more embodiments of the invention, picture box viewers 220-228 are web browsers configured to obtain picture box data from server system 215. Picture box engine 216 is responsible for generating picture boxes. To accomplish this task, picture box engine 216 obtains picture box data 233 used to create the picture boxes 229-233 from data repository 213. The invention contemplates placing picture box engine 216 in a variety of different locations. In one embodiment, for example, picture box engine 216 resides at server system 215. However, picture box engine 216 may also reside on any computer readily accessible to clients 219-227 via interconnection fabric 217.

FIG. 2b illustrates the system components utilized by one or more embodiments of the invention. Frame devices 200a and web browser 262 are both controlled by an authorized user 283. The term user as described herein has multiple meanings. The term comprises guest users who have not yet registered and/or initialized a frame device. The term user also refers to registered guests (e.g. members) that may submit images to a frame device. An owner is a type of user who has a frame device. However, an owner need not use web browser 262 to configure the frame device if a user of

another type or the manufacturer of the device already has configured the frame. One or more embodiments of the invention contemplates the use of an owner-proxy. The owner-proxy is authorized by the owner to act on the owner's behalf. For example, the owner-proxy may be given authority to control certain aspects of the frame device by the owner. A parent is the user who controls one or more frame devices. The parent has the authority to control what type of images is displayed on the frame and who may submit images for display. The parent user typically has all the privileges needed to control the frame device. The purchaser or buyer of a frame device may also be referred to as a user. A person who provides images and/or data to the system such as a gallery provider, channel provider, and/or advertisement provider is also considered a user of the system. Thus, the term user is intended to be a general term that has multiple meanings. A single person or entity may function as multiple users and multiple people may function as one or more users. The invention contemplates assigning each type of user a different password. Thus, each user may be assigned a different level of access to the system. For example, a parent user has permission to modify the picture box in any way whereas a member does not yet have permission to modify the frames behavior, but may submit images to the picture box for display on the frame device.

When a user initially connects frame device 200 to a power source and/or a communication source, the frame device interfaces with telco 278 via telecommunication mechanism 263. Telecommunication mechanism 263 connects to access array 264. In one or more embodiments of the invention telco 278 is the infrastructure provided by a telecommunications provider (e.g. AT&T, MCI, SBC, PacBell, GTE, etc. . . .) and telecommunication mechanism 278 is a local phone service and/or toll-free phone service. Access array 264 is provided by ISP 279, and comprises one or more devices configured to communicate with the carrier wave signals transmitted by frame device 200. The invention contemplates the user of multiple access arrays and each access array may be located in a different geographic region. Each access array may be configured to understand the signaling and/or communication protocols utilized by the frame device. For example, if frame device 200 uses a modem to connect to access array 264, access array 264 may be configured to understand the corresponding signaling standard (e.g. V.32, V.42bis, V.90 etc. . . .). Access array 264 may also comprise computer hardware configured to understand protocols such as TCP/IP, SLIP, and PPP. The services offered by MegaPOP™, ZipLink™, UUNET™, and/or GTE Internetworking™, for example, are examples of an access array 264.

Access Array 264 communicates with proxied server(s) 266. Proxied server 266 may be, for example a RADIUS server configured to communicate with the registration server 268. RADIUS is a program that runs on a computer. RADIUS allows the expansion of the number of user (connection) profiles available to RADIUS compliant remote access devices. The three main functions of a RADIUS server are authentication, authorization, and accounting. RADIUS is used to authenticate users for dial-in and/or remote access via a data communication network. Authentication information may be stored locally (e.g. in a local file) or accessed from external authentication mechanisms. Authorization controls access to specific services on the network. Once a user is authenticated, RADIUS identifies what a user is authorized (permitted) to access and communicates that information to a PortMaster. For example, user _1 may be authorized to use a certain type of

17

communication protocol (e.g., PPP or SLIP) for his connection, and to use a certain IP address. The RADIUS accounting permits system administrators to track dial-in and/or network use. The present invention is not limited to the use of RADIUS servers, but also contemplates the user of other types of servers that perform the functionality required to effectively balance the communication path between frame device 200 and data repository server 269.

In one or more embodiments of the invention, access array 264, proxied server 266, load balancer 267, and registration server 268 intercept and then forward data signals transmitted from frame device 200 to data repository server 269. Load balancer 267, for example, is responsible for balancing the load placed on devices in the communication path (e.g., registration server 268). A Domain Name System (DNS), Windows load balancing service, Router, or Round Robin device are some examples of systems capable of acting as load balancer 267. Load balancer 267 is provided by frame data provider 281 to communicate with data repository server 269 via registration server 268. The functionality provided by load balancer 267, 272, and 276 may be provided by a single computer or multiple computers. A single computer connected to the interconnection fabric may, for example, be configured to execute software or hardware that acts as load balancer 267. Alternatively, the functionality provided in load balancer 267 may be executed in other computer systems connected to the interconnection fabric (e.g., Registration Server 268 or Data Repository Server 269).

Data repository server 269 and registration server 268 may be configured to transmit log information 274 to one another. In one or more embodiments of the invention log information 274 is RADIUS logs. Data repository server 269 executes, manages, and communicates with data repository 270 via Local Area Network (LAN) 273. Data repository 270 is a database file system and image store. However, data repository 270 may also comprise other information needed to update and/or manage frame device 200. For example, the data contained in Package Server 271 may be stored in data repository 270.

Package Server 271 is configured to communicate with data repository 270 and data repository server 269 via LAN 273. In one or more embodiments of the invention, package server 271 comprises a File Transfer Protocol Daemon (FTPd), a packager, and an interpreter. Package server 271 may also be referred to as an application server. The packager is responsible for generating multiple kinds of packages. A package is a compilation of data that provides the recipient and the transmitting device with information needed to complete a transaction. For example, a package may contain authentication information that provides frame device 200 with a way to verify whether package server 271 is authentic. A package may also contain metadata about frame device 200. For example, some or all of the information stored in flash memory and/or DRAM may be inserted into a metadata file. The metadata file may contain, for example, a unique frame identifier, a relative clock time, a lights on clock tick, a lights out clock tick, a connect time, connection information, slide show information, log information, name server information, image information, an image display list, and error information. Other information utilized by the frame device and/or any of the server systems may also be placed into the metadata file. The package server 271 may also generate packages that contain content and formatting data (e.g., image and/or text data).

To obtain package data frame device connects to access array 264 via telecommunication mechanism 263. In one or

18

more embodiments of the invention frame device 200 is configured to establish an FTP session with package server 271. Once the session is established, frame device 200 transmits a unique identifier (e.g., user/frameID). Package server 271 responds by prompting frame device 200 for password information, which the frame then transmits. The frame device may dynamically generate the password information. Once frame device 200 enters the correct password, the device is permitted access to package server 271. At this point frame device 200 is linked to package server 271 and data may be sent to and from each using either ASCII mode or binary mode. Load Balancer 272 may be optionally placed between package server 271 and access array 264 in order to ensure that the load placed on package server 271 does not exceed its capacity.

Frame device 200 typically communicates with package server 271 using the binary connection mode. During the communication session frame device 200 initiates a GET command via path and thereby obtains an authentication file from package server 271. When frame device 200 obtains the authentication file, it determines if the authentication file is authentic by computing a checksum and comparing the result to an expected results file. If the comparison matches, frame device 200 "trusts" package server 271. When a trust relationship is established frame device 200 transmits package data to package server 271. In one or more embodiments of the invention, frame device 200 tells package server 271 how much package data it will be sending (e.g., via a SIZE command) and then transmit that amount of data to package server 271 via a PUT command. For example, a content list and a log file may be transmitted to the server via the PUT command. Package server 271 then computes and compares the checksum information associated with the transmitted package data (e.g., metadata). Frame device 200 may also utilize the GET command to obtain additional packages. In one or more embodiments of the invention, the USER command and PASS command are modified to provide an additional layer of security. For example, the PASS command and the USER command may be modified to transmit and/or bounce back encrypted strings of data that are authenticated before access to the package data is permitted.

For example, frame device 200 may use the GET command to obtain an administrative package comprising information useful for administering the frame and/or a content package comprising content and formatting data. The content of the packages is self-defined and may vary depending upon the requirements of frame device 200. Upon delivery of the relevant packages, frame device 200 may synchronize its time keeping mechanism with package server 271 and terminate the connection. If the frame device connected using the toll-free phone number, local number information is obtained and provided to the frame device in the administrative package. The local number information that is best for the device is determined by analyzing the caller ID information provided when the device call using the toll-free phone number.

In one or more embodiments of the invention, web browser 262 provides a mechanism for submitting and/or changing the image data that is to be displayed on frame device 200. To submit image data web browser 262 connects to local ISP 265 via telco 278. Local ISP 265 communicates with load balancer 276 which in turn communicates with one or more web servers 275 and submission/image processing engine 284. Web browser 262 and web servers 275 may also communicate with clearing house 277 and/or data repository 270. Clearing house 277 is a computer system configured to process transactions with user 283. In one or

19

more embodiments of the invention, submission/image processing image comprises software configured to obtain image data from user 283 via web browser 262.

The Data Repository

The data repository comprises one or more databases. Each database provides the system with information and/or data related to distributing data to one or more frame devices. In one or more embodiments of the invention, the data repository stores information used to control the content distributed to and from each frame. For example, data about each user and the preferences associated with that user may be held in the data repository. Each frame device is configured to connect to the data repository at one or more predefined intervals utilizing an interconnection fabric such as the Internet. Once a frame device connects to the data repository, it utilizes the information stored in the data repository to update the image data that is to be displayed and the behavior characteristics of each frame device. For example, the functions provided by the onboard software may be modified when the behavior characteristics are updated. The data repository is therefore responsible for queuing and archiving image data and/or software data for each frame device associated with the data repository.

Referring now to FIG. 4, an example, of the type of data stored in the data repository is shown. In one or more embodiments of the invention, the data repository is comprised of multiple databases. The databases are related to one another and contain information utilized to generate the data repository. Each database may contain one or more tables of information. Thus, the data repository may be constructed as a single database having one or more tables or as a series of databases related to each other. The term database and table may therefore be used interchangeably. The databases may be organized using a relational structure, an object-oriented structure, or any other type of organizational scheme configured to store the information needed to create the data repository. The databases may also be distributed across multiple computers. For example, a first computer may contain one portion of the data repository and a second computer may contain a second portion of the data repository.

One of the databases contained in the data repository is the user database 401. User database 401 comprises information about users that have registered with the system. In one embodiment of the invention user database 401 contains one record for each registered user. User database 401 typically stores information about the user such as the user's username, password, picture mail address, and billing information. User database 401 may also contain a list of picture mail addresses from which image data may be received. If, for example, a user wishes to receive picture mail from a certain address the user may designate that address as a friendly sender by placing the address in the list of picture mail addresses from which data may be received. In one or more embodiments of the invention, picture mail addresses not contained in the list may not populate a user's frame device with image data. The user may use the picture box to identify a list of friendly picture mail addresses. This list of friendly senders may be referred to as a buddy list and/or address book.

The data repository also holds information about each frame initialized with the system. A frame becomes initialized when the user of the frame device, or a person acting on behalf of the user, directs the system to obtain a picture box. Information about the frame device is held in the frame

20

database 403. The frame database 403 comprises multiple records 404-411*n* and each record is associated with a unique frame device. A single user, however, may have multiple frames registered with the system. Thus, one record in user table 401 may be related to many records in the frame table. For example, if a user owns two frame devices the data repository may have a single record about that user in user database 401 and two records about the user's frame devices in frame database 403. Frame database 403 contains a record for each frame device that registers with the system.

Frame database 403 contains information that uniquely identifies each frame. Frame database 403, for example, may store a serial number or any other form of information that uniquely describes a particular frame device. Each user, may control one or more frame devices. In one or more embodiments of the invention, frame database 403 stores information about the geographical location of each frame. For example, the zip code, the area code, and/or phone number where the physically device resides, or any other information that identifies the location of the device may be used to describe the geographic location of the frame.

In one or more embodiments of the invention, each frame is associated with an access number. The access number is the phone number utilized by a particular frame device to connect to the interconnection fabric. The access number, for example, may be a toll free phone number or the phone number of an ISP closest in proximity to the physical location of the frame device. The access number may be stored in frame database 403.

Frame database 403 also contains data that identifies the intervals at which a particular frame device may perform a predefined action. For example, frame database 403 may hold information that specifies the date and time the frame device associated with that record is to connect to the data repository. The actions performed may be defined by the user of the frame device or by an entity authorized to control the actions of the frame device. One embodiment of the invention allows any person with access to the picture box to define an action.

The intensity of each frame device's display region may be varied. For example, the display of each frame device may be configured to dim at certain times. The dimming times are stored in frame database 403. For example, if the user wishes to have a particular frame device dim between the hours of 11:00 p.m. and 6:00 a.m. this information may be specified via the picture box and is stored in frame database 403. The frame device may also be configured to automatically dim at a certain time based on the longitude and latitude of the device. The system obtains the location of the frame device when the device indicates it is present by calling the toll free phone number embedded in memory. The caller ID information passed during this connection, for example, allows the system to determine the location of the device and set the corresponding dimming times. The frame device is typically directed to dim during times when network traffic is low and connect time is inexpensive, however, dimming may occur at any time determined to be appropriate by the system. For example, dimming may occur at a time that is not associated with the geographical location of the frame device.

Information about the software contained in each frame may also be stored in frame database 403. For example, frame database 403 may store the release and/or version number of the software used to operate and control each frame. If the software is not the most current version available an indication as to whether it is ok to download the

software may also be stored in frame database 403. Frame database 403 stores information about the connection history of each device. For example, the number of failed connections, the number of successful connections, as well as the speed and duration of those connections is stored as the connection history. The connection history may also contain a record of the communication sessions established by a particular device (e.g. the phone numbers called by a particular frame device and the number of images transmitted to the device at that phone number). The frame database 403 may also contain a list of the administrative information uploaded to each frame device.

In one or more embodiments of the invention, the data repository also contains content provider database 413. A content provider is an entity or organization that offers one or more image delivery services via the interconnection fabric. Content provider database 413 may contain information about multiple content providers. Content provider database 413 contains the name and location of each content provider. For example, content provider database 413 may store the web address or Internet Protocol (IP) address where data pertaining to a particular image delivery service may be located. Content provider database 413 may also contain pointers to other databases where image data utilized to offer an image delivery service resides.

The data repository also stores data utilized by multiple frame devices. For example, central administration database 415 comprises information such as the time of day (per time zone), administrative messages and/or images, broadcast messages and/or images, and other general data utilized by the frames devices when a connection to the data repository is made. An example of a relational database schema arranged to store the data utilized by the system is shown in appendix A.

The Picture Box/Picture Mail Community

When a user initializes a frame device, that frame device becomes associated with a picture box. For example, each frame device 200-209 is uniquely identified (e.g. by a serial number, address, digital signature, or other identifier) and associated with a particular picture box (e.g. an account at a web site). Each picture box is associated with a unique username and/or picture mail account. Once a picture box is obtained, the user associated with the picture box becomes a member of a picture mail community. Members of the picture mail community are each provided with one or more picture mail addresses. A picture mail address provides a mechanism for transmitting image data to a frame device. The picture mail address, for example, provides the system with the addressing information needed to identify the location of the frame and transmit image data to the frame. Thus, the picture mail address provides user with a way to pass images between other users (e.g. family and friends) who also belong to the picture mail community. In one or more embodiments of the invention, a picture mail address is a network address that uniquely identifies the location of each frame device. The picture mail address may be implemented in the form of an electronic mail address, an IP address, or any addressing scheme that informs the system how to direct image data to the frame device.

Image data may be sent to the frame device by sending the data from a data source (e.g. a client or server computer) to the picture mail address associated with that device. The picture box provides the user with an interface for managing the behavior characteristics of one or more registered frame devices. In one or more embodiments of the invention, each

user is given a single picture box to control the behavior characteristics of multiple frame devices. However, a user may also be given multiple picture boxes. The picture box is accessible via an interconnection fabric such as the Internet and may be viewed using a standard web browser. In one or more embodiments of the invention, each frame device is associated with a specific picture box. Information about the picture box utilized to control a frame device may be stored in frame database 403.

Information used to generate each of the picture boxes may be stored in picture box database 400. For example, picture box database 400 contains a representation of the image (e.g. a thumbnail) as it will be displayed on the frame device, should it be flagged for download to the device. The location (e.g. picture mail address or network address) the image data was transmitted from is also stored in picture box database 400. Other information such as the date and time the image was received at the data repository and the date and time the image was received at a particular frame device also resides in picture box database 400. Picture box database 400 may also contain information about the status of the image data accessible to a particular frame device, such as whether the image is to be archived or kept on the frame device when an update occurs. Data that specifies the filename of the image, whether the image currently resides on the frame device, and routing information that specifies which images should be sent to which frame devices may also be stored in picture box database 400.

FIG. 5 shows an illustration of a picture box viewer containing a picture box. In one or more embodiments of the invention, picture box viewer 500 is a web browser configured to display picture box 523 (e.g. a web page). Examples of several widely used web browsers include Netscape Navigator, Internet Explorer, and Opera. However, picture box viewer 500 may be any type of software program configured to display picture box 523. Picture box 523 is any type of data file cable of being transmitted to picture box viewer 523 for display. For example, files written using the HyperText Markup Language (HTML), the JavaScript programming language, the ActiveX programming language, or the Portable Document Format (PDF) may be displayed using picture box viewer 523. It is also possible to generate picture box 523 using various other types of languages such as Standard Generalized Markup Language, (SGML) or extensible Markup Language (XML).

Picture box 523 is any example of the type of interface that may be used to create and/or modify records in the data repository. The information stored in the data repository may be presented to the user via picture box 523. In one or more embodiments of the invention, picture box 523 comprises a first portion 505 that contains a list of the image delivery services scheduled to provide data to a particular frame device and a second portion 509 that shows the status of images that may be displayed on the frame device. However, picture box 523 is not limited to using first and second portions, but may utilize any number of portions to present information to the user of a frame device.

In one or more embodiments of the invention, first portion 505 shows what image delivery services the frame device associated with picture box 523 is subscribed to. For example, the frame associated with picture box 523 is scheduled to receive weekly subscription 503, daily subscription 505, and monthly subscription 507. The delivery intervals may be changed utilizing picture box 523. For example, it is possible to use picture box 523 to change weekly subscription 503 to a monthly subscription.

The second portion 509 shows what image data is currently intended for display frame devices associated with

picture box 523. Second portion 509 comprises columns of information likely to be of use to the owner of the relevant frame device. For example, in one or more embodiments of the invention, columns that contain image name 512, status 511, date 513 and sender 514 are shown in second portion 509. However, any type of data that describes the images currently accessible to the frame device may be placed in the image name 512 column. For example, the image name 512 column may contain the actual file name, an alias associated with the image file, or a description of the image itself. Image name 512 may also contain a representation of the image data such as a thumbnail.

The status 511 column comprises information that represents the status of the image. For example, information that communicates whether an image is scheduled for display on the frame device is shown in status 511 column. Other information such as whether a new image was received, or whether the image may or may not be deleted can also be presented using status 511 column. One embodiment of the invention provides for the placement of any data that communicates the location and/or status of an image in status 511 column.

Picture box 523 may be configured to show a date 513 column. The date and/or time an image was received may be placed in date 513 column. Other data that is relevant to the receipt of an image may also be inserted into date 513 column. For example, the date a particular image was loaded into an associated frame device may also be placed in date 513 column.

Sender 514 column shows who sent the images shown in the picture box. In one or more embodiments of the invention, sender 514 column contains the picture mail address of the party who transmitted the image. The name and/or location the image was transmitted from may also be shown in sender 514 column.

Routing information may be also be defined using picture box 523. For example, if the user has the authority to control what is shown on multiple frame devices, the user may use the picture box to specify which images should be sent to which frame devices.

Picture box 253 also provides a mechanism for users to browse through the image data available for display. For example, users may search for images in a list of galleries by conducting a key word search of the galleries. Images held, in the galleries may be obtained from a person or entity, that provides stock images to the picture box. The user may obtain thumbnail views and full size views of the image data upon request. The user may also use the picture box to review all past, present, and future images displayed on the frame device. Users can also use the picture box to add, replace, or delete any image that is to be scheduled for display on the frame device. Thus, the picture box provides an interface for remotely controlling the behavior of the frame device.

The Image Delivery Service

In one or more embodiments of the invention, users may elect to subscribe a particular frame device to one or more image delivery services offered by the content providers. When the user subscribes to an image delivery service, customized image data is periodically delivered to the frame device designated by the user as the recipient. For example, if a user wishes to have an image summarizing the weather report regularly delivered to the frame device located in the user's house, the user may subscribe to an image delivery service that regularly transmits weather images to the picture

mail address associated with the appropriate frame device. The user may control the date and time the image delivery service sends the image data. The user, for example, may specify that the weather report is only to be delivered at 7:00 a.m. on weekdays and is not to be delivered during the weekend. The invention also contemplates a system wherein the data and time the image delivery service transmits image data is predefined and not set by the user. For example, the image provider may determine when to send images to the picture mail account associated with a frame device. A number of different image delivery services are available to members of the picture mail community and each delivery service may transmit images that are illustrative of any kind of information. For example, images that represent post cards, greeting cards, art collections, live video feeds, or any other type of image data may be delivered to a frame device utilizing an image delivery service.

Frame Device Initialization

In one or more embodiments of the invention, the frame device initializes with the system in order to have the capability to obtain image data. The initialization process may occur with the aid of a user or it may occur automatically without user input. In either instance, users are given the opportunity to customize the behavior of the user's frame device(s). FIG. 8 illustrates the process utilized by one embodiment of the invention to initialize a frame device. The initialization process begins at step 800 when the frame device is connected to a power source and a communication medium such as a telephone line or network connection. The frame device may be configured to automatically connect via a wireless connection or the user may manually couple it to al power source and communication medium.

Once step 800 occurs, an embodiment of the invention proceeds to step 803. At step 803, the frame device initiates a connection to the data repository utilizing an interconnection fabric. In one or more embodiments of the invention, the frame device is configured so that can only connect to the data repository and cannot connect to other data sources that are available via the interconnection fabric. For example, the connection process may utilize an authentication scheme (e.g. encryption) to ensure the device connects to an authorized data repository and is not being directed to another data source masquerading as the authorized repository. When step 803 is complete, step 805 executes. At step 805, the data repository is analyzed to determine whether there is a record for the frame device initiating the connection.

If a record for the frame device is not located, step 807 executes and an image prompting the user to create a picture box account associated with that frame device is displayed to the user. Once the picture box account is created it provides the user with a way to set the behavioral characteristics of the frame device. In one or more embodiments of the invention, the frame device cannot properly connect to the data repository until the picture box account is generated. Thus, the invention contemplates generating a picture box by executing steps 811 and 813.

At step 811, a picture box is obtained. The user may obtain a picture box, and thereby join the picture mail community, by calling a toll free phone number to have another person initiate the generation of the picture box. Alternatively, the user may obtain a picture box by using a web browser and connecting to a web site that contains a mechanism for generating the picture box. The frame device may also automatically direct the system to generate a picture box if an attempt is made to connect and a picture

25

box associated with the frame device attempting to make the connection does not exist. This may occur without requiring any input from the user. However, user input may be obtained if such input is deemed to be desirable. In one or more embodiments of the invention, however, the user may personalize the picture box by using the input mechanisms present on the frame device (e.g. the pushbutton switch).

Once the picture box is obtained, step 813 executes, at step 813 the behavior characteristics of the frame device is determined. The behavior characteristics may be established by default or customized according to the user's preferences. When the behavior characteristics are determined, step 815 executes. At step 815, the invention contemplates loading the behavior characteristics into the data repository for use by the frame device.

If the data repository already contains a record for the frame device initiating the connection, step 809 executes. At step 809, the frame device obtains the behavioral characteristics set via the picture box, by accessing the data repository. One embodiment of the invention loads the behavioral characteristics into flash memory residing on the corresponding frame device. Other data, such as software programs and/or image data may also be loaded into the frame device during step 809. Thus, functionality may be added to the software contained in the frame device without input from the user.

Self-Configuration Protocol

FIG. 9 illustrates another process that may be utilized by one or more embodiments of the invention to initialize a frame device. The initialization process begins at step 900 where the frame device determines if it is connected to a power source and a communication medium such as a telephone line or network connection. If so, the frame device proceeds to step 901 where it obtains image data from its own memory. At step 902, the image data is displayed to the user via display 313. In one or more embodiments of the invention, the image data directs the user to proceed to step 903 where the frame device manually initiates a communication session. The frame device may be configured to automatically initiate a communication session within a certain interval without requiring input from the user. For example, if the user connects the frame device to power source and a communication source, the frame device may initiate a communication session if the user does not manually do so within a certain time frame.

At step 903, the frame device determines whether it has local phone number information in resident memory. For example, if the frame device has not previously initiated a communication session the device will not have any local phone number information. If so, step 904 executes. At step 904, the device obtains a phone number (e.g. toll free phone number) from the memory of the frame device. This phone number is referred to as a configuration number sequence and may be permanently embedded into persistent memory (e.g. nonvolatile memory). The configuration number sequence is used only for configuration access and is parsimonious with the amount of time it remains connected. Once the frame device has the configuration number sequence, it dials the configuration number using a tone sequence (e.g. step 905). If that fails, the device may use pulse mode. When a connection is established, device proceeds to step 906 where it obtains one or more local phone numbers from a server computer (e.g. the data repository) provided by the frame service provider. The local phone number information is stored in memory at step 907.

26

Once step 907 executes, the device proceeds to step 908 where it disconnects from the configuration number and re-dials using one of the local numbers. If the frame device already has obtained a local number, step 909 executes and when the device is directed to initiate a connection it dials the local number stored in memory. Thus, the frame device is capable of automatically connecting to the interconnection fabric where it may obtain image data from the data repository without requiring input from the user. If the local phone number information is flushed from memory (e.g. by a power failure) or becomes inaccurate (e.g. because the user moved the physical location of the device), the device may use the toll free phone number to automatically reconfigure itself without prompting the user for additional input. Thus, each frame device is an intelligent device that can be easily maintained by novice or inexperienced users.

Image Collection/Submission Engine

FIG. 6 is a flow chart illustrating the process utilized by one embodiment of the invention to prepare an image for display in the display region of a particular frame device. At step 600, the user wishing to transmit the image data or a data collection engine, obtains the data and prepares it for submission to the picture mail address associated with a particular frame. The data repository may act as an image relay mechanism which holds the image data until the frame device connects to the repository for an update.

The data may be obtained from any source capable of providing one or more image files. For example, image data may be collected from a client computer, an image scanner, digital camera, or a memory device such as a hard drive. The data collection engine provides a mechanism for obtaining images from any image source accessible via an interconnection fabric such as the Internet. In one or more embodiments of the invention, the data collection engine is configured to obtain image data from any web server or web client connected to the interconnection fabric. For example, the user may direct the data submission engine to periodically collect one or more images from a networked device (e.g. a web server or series of web servers) by specifying a URL that identifies the location of the networked device containing image data that is to be collected.

At step 603, the image data is prepared for submission. An embodiment of the invention contemplates examining the image data to discern whether it meets a set of predetermined constraints. For example, if the display area of the frame device is a static width and height (e.g. 640x480 pixels), step 603 will obtain the dimensions of the display area and determine whether the image data will fit within the identified area without alteration. If the image does not fit, the image data may be altered to coincide with the dimensions of the frame device. The alteration process may scale the image, crop the image, change the resolution of the image, or otherwise modify the image so that it fits within the given display area. For example, if an image is 1024x768 pixels and the display area of a particular frame device is 640x480 pixels, the image may be appropriately scaled to fit into the 640x480 display area of the frame device. In one or more embodiments of the invention, step 603 is performed before the image data is transmitted to the data repository. However, software executing at the data repository or at the frame device may also be configured to prepare the image data for submission. One embodiment of the invention contemplates placing software configured to perform step 603 on any computer connected to the interconnection fabric.

In one or more embodiments of the invention, the data format of the submitted image is modified to conform to the

preferences of the frame device. For example, if the submitted image is in BMP format and the frame device contains software or hardware that is capable of rendering JPEG images for display, the image will be converted from BMP format to JPEG format to coincide with the needs of the frame device. The dimensional aspects of the image may be altered along with the data format. For example, a BMP image having a resolution of 1024x768 may be converted into a JPEG image having a resolution of 640x480. The color depth and other aspects of the image may also be modified. For example, the image alteration process may convert all JPEG images so that each image submitted to the system has a 12-bit color depth and is compressed to a size that does not exceed a certain size (e.g. 64 Kb). Submitted data is conformed to coincide with size and format parameters in order to prevent large, illegal, or improperly formatted files from consuming resources on the system. Additionally, filtering and modifying images before they are stored in the data repository provides a way to prevent excessively large images from consuming resources on the frame device. The data format conversion and the image resizing process may be performed without requiring input from the user. However, in one or more embodiments of the invention, the image alterations may be presented to the user for approval and/or manual modification.

When a user initializes a frame device, that frame device becomes associated with a picture box. Each picture box is associated with a unique username and/or picture mail account. At step 605, the username and/or picture mail account information is utilized to provide a mechanism for transmitting image data to a frame device or series of frame devices. For example, if a user named user_1 initialized frame device XYZ with the system, another user could transmit image data to frame device XYZ by transmitting the image data to the picture mail address associated with user_1. A standard electronic mail client or a customized picture mail client may be utilized to transmit the picture mail and its associated image data. However, any interface capable of sending binary image data to a particular network address (e.g. a web-based interface) may also be utilized.

At step 607, the image data is analyzed to determine if it meets the filter criteria established by the user during the registration process (e.g. is the submitted image data from a person who belongs to the user's buddy list or, address book). In one embodiment of the invention step 607 is performed by a data population engine. The data population engine utilizes a specific set of filter criteria to determine whether an image is appropriate for display on a frame device. An embodiment of the invention provides the user with an interface to set the criteria. For example, the filter criteria may be set using the picture box associated with that user. The user can, for example, prevent images arriving from certain picture mail and/or network locations from being displayed on the user's frame device(s).

The user may also elect to store images that arrive from certain locations in the data repository and not present the images for display until they are reviewed and/or approved by the user. The user or any other authorized individual and/or entity may set the filter criteria. For example, the organization that controls the data repository may be given the authority to establish filter criteria. The user may set the filter criteria used by the data population engine using the picture box interface. Other interface devices capable of receiving input from a user, such as a telephone or Personal Desktop Assistant (PDA), may also be used to set the filter criteria.

When an image satisfies the filter criteria step 609 executes. At step 609, images are loaded into the appropriate

portion of the data repository. For example, image data that is transmitted to the picture mail address of user_1 may be loaded into the record in the data repository created for user_1. Each time another image is submitted it is added to the appropriate record. Thus, the data repository may hold one or more images.

If the image data does not meet the filter criteria, the image data is discarded or returned to the address of the sender at step 606. In one or more embodiments of the invention, data that does not meet the filter criteria is stored in the data repository, but is not forwarded to a frame device until approved by the user.

Once image data resides in the data repository, step 611 executes. At step 611, the frame device obtains the image data set residing at the data repository. For example, the frame device may connect to the data repository utilizing the interconnection fabric and download the image data into memory. In one or more embodiments of the invention, the frame device periodically connects to the data repository. The frequency of the connections may be determined by the user via the picture box interface or by a party authorized to access the picture box. The device may also automatically determine the connection frequency without requiring input from the user.

Once the image data resides in flash memory, step 613 executes and the image data is displayed on the frame device's display mechanism according to the display preferences expressed via the picture box.

Conversation/Security Protocol

FIG. 10 illustrates the process used by one or more embodiments of the invention to obtain new image data and/or software updates. At step 1000, the frame device establishes a connection to the access array at the scheduled interval via the interconnection fabric. For example, the frame device may utilize the communication interface to connect to an ISP or other entity with an access array such as a modem pool. The frame device obtains the information it needs to initiate the connection from flash memory (e.g. phone number information, unique identifier such as a serial number, and password information). Other information such as the network router information, subnet mask, and domain name service information is drawn directly from the ISP using DHCP or some other dynamically addressing protocol. Thus, the device may obtain the information it needs to connect without input from the user. Once the connection is initiated, the process proceeds to step 1002 where the ISP or entity having control of the access array determines if the frame device is authorized to connect to the access array. If for example, the frame device connects using a modem and PPP, the access array may prompt the device for password information that may be automatically entered by the frame device. In one or more embodiments of the invention, the password to be entered is determined algorithmically by the frame device. If the correct password information is entered the process proceeds to step 1004, otherwise step 1003 executes and the frame device disconnects and waits for a reconnect command to be initiated.

At step 1004, the frame device establishes a communication session with a data server. This may be done, for example, using FTP to connect to the data server. However, the invention contemplates the use of other communication protocols such as NFS, iFTP, HTTP, SMTP, POP3, IMAP, or any other protocol that can be used to transport data between two or more sources, to establish a communication session with the data server.

When the communication session is initiated the data server executes step 1006 where a challenge is made to ensure the frame device is authentic. If the frame device correctly responds to the challenge, step 1008 executes. Otherwise, step 1003 executes and the frame device disconnects. At step 1008, the data server transmits data to the frame device. For example, the data server may enter binary mode and transmit an authentication file to the frame device in response to the device's request for data (e.g., via the GET command). Once the data is sent step 1010 executes. At step 1010, the frame device examines the data it received to determine if the data is the type of data expected. For example, the frame device may authenticate the data by checking the size, contents, or encryption sequence associated with the data to determine if the transmitted data came from an authorized data server. If the frame device received the expected data (e.g., the encryption sequence is authentic), it proceeds to step 10012. If the frame did not receive the kind of data expected, step 1003 executes and the frame disconnects from the data server and awaits a reconnect command.

At step 1012, the data server transmits metadata and/or a log file to the data server. The metadata file contains status information about the frame such as configuration information or connectivity information. The metadata file may also contain new or additional parameters and/or functionality that are to be added to the device (e.g., an on board software update). The log file contains a record of the frame devices past activities. For example, the log may contain a list of recent connection attempts. Any connection failures may be flagged and later analyzed to determine the cause of the failure. After the frame device transmits the metadata to the frame device step 1014 executes. At step 1014, the data server determines if it received the kind of metadata that was expected. For example, the data server may check to see if the metadata file contains all the expected parameters. In one or more embodiments of the invention, the data server checks the received data to ensure that the data is the proper type of data and that it has not been modified by an unauthorized user (e.g., by examining an encryption sequence associated with the metadata). If the metadata is properly authenticated, the data server executes step 1016 where it transmits package data to the frame device. The package data may contain one or more images that are scheduled for display on the frame device, software updates, and additional parameters needed to direct the behavior characteristics of the frame device. Other information that is useful to the operation of the frame device such as timing information may also be placed in the package. Once the package is received by the frame device step 1018 executes. At step 1018, a checksum operation is performed to ensure that the entire package scheduled for transmission was indeed transmitted. If the checksum results in the expected response the frame device disconnects and begins to unpack the package data. If the package data is compressed, the frame device decompresses the package and loads the contents of the package into memory. Image files found in the package may be stored in DRAM until they are rendered for display.

Embodiment of Computer Execution Environment (Hardware)

An embodiment of the invention utilized to access the picture box interface can be implemented as computer software in the form of computer readable code executed on a general purpose computer such as computer 700 illustrated in FIG. 7, or in the form of bytecode class files executable

within a Java™ runtime environment running on such a computer, or in the form of bytecodes running on a processor (or devices enabled to process bytecodes) existing in a distributed environment (e.g., one or more processors on a network). A keyboard 710 and mouse 711 are coupled to a system bus 718. The keyboard and mouse are for introducing user input to the computer system and communicating that user input to processor 713. Other suitable input devices may be used in addition to, or in place of, the mouse 711 and keyboard 710. I/O (input/output) unit 719 coupled to system bus 718 represents such I/O elements as a printer, A/V (audio/video) I/O, etc.

Computer 700 includes a video memory 714, main memory 715 and mass storage 714, all coupled to system bus 718 along with keyboard 710, mouse 711 and processor 713. The mass storage 714 may include both fixed and removable media, such as magnetic, optical or magnetic optical storage systems or any other available mass storage technology. Bus 718 may contain, for example, thirty-two address lines for addressing video memory 714 or main memory 715. The system bus 718 also includes, for example, a 64-bit data bus for transferring data between and among the components, such as processor 713, main memory 715, video memory 714 and mass storage 714. Alternatively, multiplex data/address lines may be used instead of separate data and address lines.

In one or more embodiments of the invention, the processor 713 is a microprocessor manufactured by Sun Microsystems, Inc., such as the SPARC™ microprocessor, or a microprocessor manufactured by Motorola, such as the 680X0 processor, or a microprocessor manufactured by Intel, such as the 80X86, or Pentium processor. However, any other suitable microprocessor or microcomputer may be utilized. Main memory 715 is comprised of dynamic random access memory (DRAM). Video memory 714 is a dual-ported video random access memory. One port of the video memory 714 is coupled to video amplifier 716. The video amplifier 716 is used to drive the cathode ray tube (CRT) raster monitor 717. Video amplifier 716 is well known in the art and may be implemented by any suitable apparatus. This circuitry converts pixel data stored in video memory 714 to a raster signal suitable for use by monitor 717. Monitor 717 is a type of monitor suitable for displaying graphic images.

Computer 700 may also include a communication interface 740 coupled to bus 718. Communication interface 740 provides a two-way data communication coupling via a network link 741 to a local network 744. For example, if communication interface 740 is an integrated services digital network (ISDN) card or a modem, communication interface 740 provides a data communication connection to the corresponding type of telephone line, which comprises part of network link 741. If communication interface 740 is a local area network (LAN) card, communication interface 740 provides a data communication connection via network link 741 to a compatible LAN. Wireless links are also possible. In any such implementation, communication interface 740 sends and receives electrical, electromagnetic or optical signals which carry digital data streams representing various types of information.

Network link 741 typically provides data communication through one or more networks to other data devices. For example, network link 741 may provide a connection through local network 744 to local computational service provider computer 743 or to data equipment operated by an Internet Service Provider (ISP) 744. ISP 744 in turn provides data communication services through the world wide packet data communication network now commonly referred to as

the "Internet" 745. Local network 744 and Internet 745 both use electrical, electromagnetic or optical signals which carry digital data streams. The signals through the various networks and the signals on network link 741 and through communication interface 740, which carry the digital data to and from computer 700, are exemplary forms of carrier waves transporting the information.

Computer 400 can send messages and receive data, including program code, through the network(s), network link 741, and communication interface 740. In the Internet example, remote computational service provider computer 746 might transmit a requested code for an application program through Internet 745, ISP 744, local network 744 and communication interface 740.

The received code may be executed by processor 713 as it is received, and/or stored in mass storage 714, or other non-volatile storage for later execution. In this manner, computer 700 may obtain application code in the form of a carrier wave.

Application code may be embodied in any form of computer program product. A computer program product comprises a medium configured to store or transport computer readable code, or in which computer readable code may be embedded. Some examples of computer program products are CD-ROM disks, ROM cards, floppy disks, magnetic tapes, computer hard drives, computational service providers on a network, and carrier waves.

The computer systems described above are for purposes of example only. An embodiment of the invention may be implemented in any type of computer system or programming or processing environment. When a general purpose computer system such as the one described executes the process and process flows described herein, it is configured to adaptably distribute data to one or more recipient devices.

Thus, a method and apparatus for distributing picture mail to a frame device community is described.

APPENDIX A

Database Tables	
Table Name	Fields
member	member_id rating_id (FK) member_nickname last_name first_name middle_initial member_email_addr address_1 address_2 address_3 city state postal_code equusity birth_date birth_month birth_year birth_day member_username member_password lock_status_id lock_status_desc picture_id (FK) lock_status_id (FK) file_name file_location pict_info pict_copyright pict_author

APPENDIX A-continued

picture_box	pict_creation_date pict_owner member_id (FK) content_id(FK) container_type_id(FK) picture_expiration_date content_id content_parent_type_id(FK) picture_id(FK)
content	content_parent_type_id(FK) content_parent_id(FK) content_parent_type
rating	rating_id rating_level channel_id
channel	channel_provider_id(FK) gallery_id(FK) rating_id(FK) channel_desc channel_update_freq channel_update_timestamp channel_provider_id(FK) channel_provider_name gallery_provider_id(FK) gallery_provider_name gallery_id rating_id(FK) gallery_provider_id(FK) gallery_name gallery_desc location_id postal_code country_id area_code location_area_offset
isp	isp_id location_id(FK) isp_name frame_id rating_id(FK) frame_parent_id(FK) member_id(FK) frame_primary_phone_id(FK) frame_secondary_phone_id(FK) frame_tertiary_phone_id(FK) location_id(FK) frame_serial_number frame_light_one_timestamp frame_light_off_timestamp frame_callback_timestamp frame_display_mode frame_user frame_pwd frame_label_number frame_software_version frame_hardware_version frame_name public_directory phone_id isp_id(FK) primary_phone_number secondary_phone_number tertiary_phone_number primary_das secondary_das frame_id(FK) content_id(FK) container_type_id(FK) slide_show_order slide_show_member picture_expiration_date picture_lock frame_queue_date frame_id(FK) history_timestamp frame_primary_phone_id frame_secondary_phone_id frame_tertiary_phone_id frame_callback_timestamp frame_software_version
frame	
frame_queue	
frame_history	

APPENDIX A-continued

frame_history_detail	frame_id(FK) history_timestamp(FK) event_timestamp event_id(FK) data_primary data_secondary
frame_events	event_id event_desc event_severity
buddy	member_id(FK) buddy_member_id(FK) buddy_name frame_parent_id(FK) member_id(FK)
frame_parent account	frame_id(FK) acct_created_date acct_status

Relationship Table		
Table name	Relationship description	Associated Table Name
member	submits may be a owns maintains may be a owns chooses owns	picture channel_provider gallery_provider frame accounts frame_parent buddy buddy picture_box member
rating	restricts ranks tasks restricts	gallery channel frame
lock gallery_provider gallery	locks provides captures	picture gallery channel
channel_provider phone frame	holds provides is distal by records	content channel frame frame_history frame_queue account
frame_history frame_event frame_parent container_type	has bills contains describes controls describes	frame_history_detail frame_history_detail frame frame_queue picture_box
picture content	fills fills	content picture_box
content_parent_type	describes	content

What is claimed is:

1. A system for distributing image data comprising:

at least one frame device configured to operate according to preferences defined by a user, said at least one frame device comprising a border region modeled to resemble a picture frame designed to circumscribe printed photographs;

a user interface coupled to at least one server system via a network wherein said user interface is physically separable from said at least one frame device and configured to obtain image data and said preferences from said user and provide said image data and said preferences to said at least one server system;

said at least one server system coupled to said at least one frame device via said network, wherein said at least one server system is configured to periodically relay said image data and said preferences to said at least one frame device when said at least one frame device automatically issues a request for said image data.

2. The system of claim 1 wherein said at least one frame device stores said preferences at said at least one frame device in at least one behavior module.

3. The system of claim 2 wherein said preferences are also stored in said at least one server system.

4. The system of claim 2 wherein said at least one frame device periodically obtains an update for said at least one behavior module by obtaining said preferences from said at least one server system.

5. The system of claim 2 wherein said at least one behavior module directs said at least one frame device to obtain said image data from a content provider.

6. The system of claim 1 wherein input to said user interface is permitted when said user is authenticated by said at least one server system.

7. The system of claim 1 wherein said at least one frame device initiates said request for said image data at intervals obtained via said user interface.

8. A method for distributing picture mail via a network to a community of frame devices comprising:

connecting at least one frame device to a network wherein said at least one frame device comprises a border region modeled to resemble a picture frame designed to circumscribe printed photographs;

obtaining a configuration number sequence from a memory located in said at least one frame device; using said configuration number sequence to initiate a connection to at least one data server via said network; obtaining a localized number sequence from said at least one data server;

terminating said connection to said at least one data server;

reconnecting to said at least one data server via said network using said localized number sequence;

presenting a user interface to a user associated with said at least one frame device, wherein said presenting executes at a location physically separable from said at least one frame device;

obtaining image data from said user via said user interface;

providing said image data to said at least one frame device via said network;

storing said image data in said memory of said at least one frame device.

9. The method of claim 8 wherein said localized number sequence is stored in said memory of said at least one frame device.

10. The method of claim 9 wherein said at least one frame device utilizes said localized number sequence when said localized number sequence resides in said memory.

11. The method of claim 8 wherein said configuration number sequence is used when said localized number sequence does not reside in said memory.

12. The method of claim 8 wherein said obtaining said image data from said user via said user interface further comprises storing said image data in at least one data repository accessible via said network.

13. The method of claim 12 further comprising: obtaining an update of said at least one frame device's onboard software from said at least one data repository via said network.

14. The method of claim 13 wherein said update to said onboard software modifies said at least one frame device's functionality.

35

15. The method of claim 14 wherein said at least one frame device determines whether said update to said onboard software is current.

16. The method of claim 15 wherein said obtaining said update of said onboard software executes when said update
to said onboard software is not current. 5

36

17. The method of claim 8 wherein said step of obtaining said configuration number sequence from said memory located in said at least one frame device occurs automatically.

* * * * *



US006058399A

United States Patent**Morag et al.**

[19]

[11] **Patent Number:** **6,058,399**[45] **Date of Patent:** **May 2, 2000****[54] FILE UPLOAD SYNCHRONIZATION**

[75] Inventors: **Guy Morag**, Kohav Yair; **Yoav Samet**,
Tel Aviv; **Leonid Entin**, Modi'in, all of
Israel; **Yoni Rosenbaum**, Portola Valley,
Calif.

5,392,422 2/1995 Hoel et al. 710/113
5,721,827 2/1998 Logan et al. 709/217
5,732,216 3/1998 Logan et al. 709/203

[73] Assignee: **ColorDesk, Ltd.**, Tel Aviv, Israel

Primary Examiner—Wayne Amsbury
Assistant Examiner—Thu-Thao Havan
Attorney, Agent, or Firm—Fenster & Company Patent
Attorneys, Ltd.

[21] Appl. No.: **08/919,862**

[22] Filed: **Aug. 28, 1997**

[51] **Int. Cl.**⁷ **G06F 17/30**

[52] **U.S. Cl.** **707/201; 707/202; 709/217;**
709/203

[58] **Field of Search** 707/201, 202,
707/203, 205; 709/217, 203; 348/7, 13;
710/113, 58, 61; 345/328

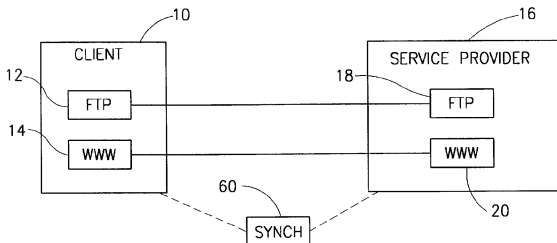
[57] ABSTRACT

A method of synchronizing an interactive connection and a non-interactive data transfer connection between a client and a service provider, comprising:

creating an interactive connection;
creating a data transfer connection; and
generating a session ID which is associated with the two connections.

[56] References Cited**U.S. PATENT DOCUMENTS**

5,319,455 6/1994 Hoarty et al. 348/7

48 Claims, 5 Drawing Sheets

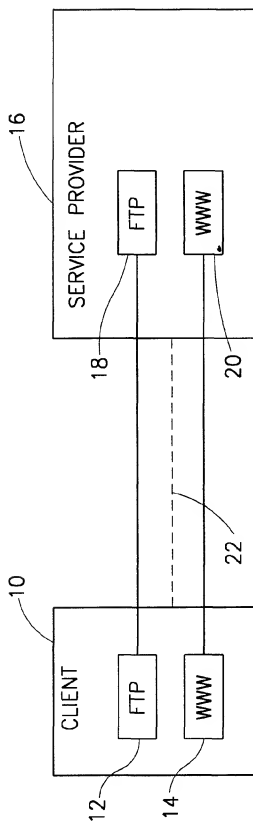


FIG. 1

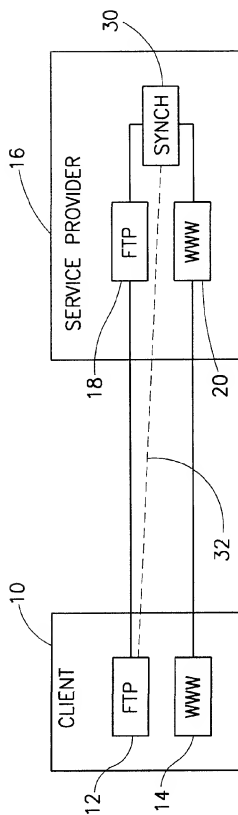


FIG. 2

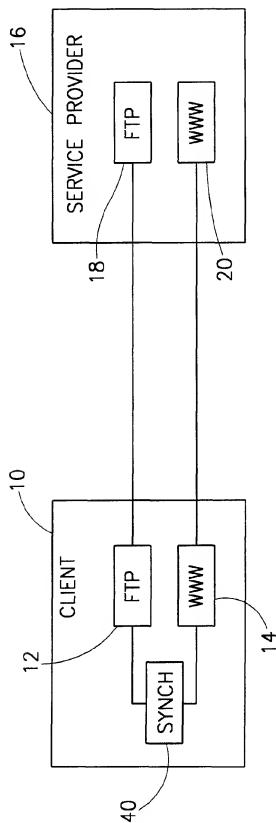


FIG. 3

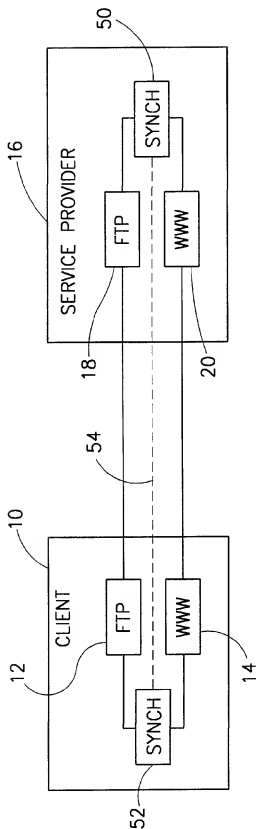


FIG. 4

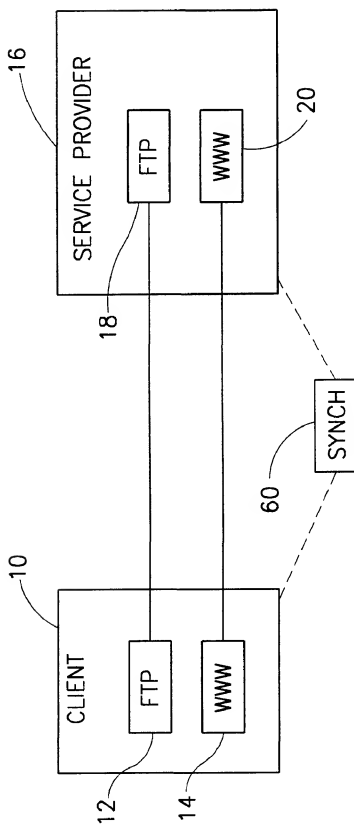


FIG. 5

FILE UPLOAD SYNCHRONIZATION

FIELD OF THE INVENTION

The present invention relates to the field of computer communications and especially to synchronizing file transfer operations with a remote interactive application, over the Internet.

BACKGROUND OF THE INVENTION

Many services can be purchased over the Internet. Some of these services only require a client to select an item from a catalog, provide payment for the item, or send other small amounts of information to the service provider. This type of information is usually sent as part of the interaction with the service provider, such as by typing. Other types of services require the client to send a considerable amount of information to the service provider. This type of information may be sent by e-mail or by uploading a file to the service provider. Files are typically uploaded either using an FTP protocol which is run independently of the WWW session, or by typing the file name(s) to a Java applet, downloaded from the service provider, as part of the session. Due to security requirements, current at this time, the file names must be manually typed for a Java applet to be able to read the files. One problem with files uploaded using an FTP protocol is that, in general, anybody may access these files.

One particular application, image manipulation, may require that large amounts of data in many separate files may be required to be sent to a service provider. All of the images which are to be manipulated/printed/developed must reach the service provider. High image quality of the final product of the service usually requires transmitting large files.

The above described solutions are cumbersome and may require a more than minimum level of computer literacy on the part of the customer. As can be appreciated, it would be desirable to provide a method of uploading large amounts of data, which method will be more user friendly than the above described methods.

SUMMARY OF THE INVENTION

It is an object of some embodiments of the present invention to provide a method of uploading large amounts of data to a service provider, which method is substantially transparent to a customer.

It is an object of some embodiments of the present invention to provide a method of uploading data which renders the data inaccessible to third parties.

It is an object of some embodiments of the present invention to provide a secure and/or convenient method of uploading data which uses a maximum number of available commercial components, both for the client and for the service provider.

A first aspect of some embodiments of the present invention relates to automatic uploading of files to a service provider. In a preferred embodiment of the invention, a customer is supplied with a software package which includes access to the service provider through a computer network. When such access is requested by the customer, the software package preferably creates two sessions, an interactive session, such as one based on a WWW protocol and a file upload session through which active data (files) are automatically uploaded to the service provider, to be used as part of the interactive session or to be processed further in accordance with choices made during the interactive session with the service provider. Active data may include a set of

data, defined as "active" by the customer or, alternatively, data which has been manipulated by the customer to a predefined extent. For example, in an image manipulation setting, all images which were previewed by the customer may be considered active, even if these images are not open anymore. Alternatively, only open images will be considered to be active. Alternatively or additionally, only images marked as active by the customer will be considered to be active. Alternatively or additionally, there are several levels of activity and once all the files in one level of activity are uploaded, the files from the next level of activity are uploaded.

A second aspect of some embodiments of the present invention relates to modifying the upload of data files responsive to the customer's interaction with the service provider. For example, if there are 10 active files, numbered 1-10, files 1-5 have been uploaded and file 6 is in the process of upload, if the customer indicates that he will only require files 4, 5 and 8, the uploading of file 6 will preferably be terminated and only file 8 will thereafter be uploaded. In another example, if a client first places an order which requires file 7, the upload order of the files will be changed so that file 7 is uploaded as soon as possible. It should be noted that should the customer delay his instructions long enough, all of the files will be uploaded. Thus, a sophisticated customer can reduce the amount of transmitted data by indicating which of the active files he wants uploaded, while an unsophisticated customer will not be required to do so. In a preferred embodiment of the invention, once the customer finishes the ordering process, any files which he did not include in an order are considered to be non-required. Alternatively or additionally, when a user cancels an order, files pertaining to that order are considered to be non-required.

A third aspect of some embodiments of the present invention relates to assuring the arrival of all the required files. One issue is that some file uploads will not succeed. In a preferred embodiment of the invention, files are automatically reloaded if their first loading does not succeed. Preferably, files which have been indicated as not being necessary for the interaction are not reloaded. Alternatively, files are reloaded only when they are indicated as necessary for the interaction. Another issue is that the customer may modify the files after they are uploaded. In a preferred embodiment of the invention, such modified files are automatically uploaded to the service provider. Alternatively or additionally, the service provider program prompts the customer for instructions whether the old or new versions of the file is desired. Alternatively or additionally, files to be uploaded are placed in a special protected directory where the customer will not have direct access to them. Another issue is that the customer may attempt to terminate an interaction before all the files are uploaded. Preferably, the service provider prompts the customer for instructions, preferably allowing the customer to cancel any of the not-yet uploaded files. Alternatively or additionally, the file upload continues even after the interaction is stopped. Alternatively or additionally, the file upload is continued at a later session. Alternatively or additionally, the missing files are e-mailed, preferably, being identified with a unique session ID. Preferably if the latter session is not upcoming, the service provider sends a notification, such as by e-mail, to the customer, reminding him of the not-yet uploaded files.

A fourth aspect of some embodiments of the invention relates to synchronizing the file upload session and the interactive session by providing a single unique ID for the two sessions. Preferably, the uploaded files are associated

with the unique session ID so that the interactive session can determine which files have been uploaded and to enable the uploading of files to be canceled via the interactive session. In a preferred embodiment of the invention, the session ID is used to differentiate multiple users and/or multiple sessions from a single user. In a preferred embodiment of the invention a single session ID is used to enable a customer to breakdown a single session into a plurality of interactive sessions. Preferably, when a customer connects to the service provider, the service provider prompts the customer to respond if he wants to continue with a previous, unfinished, session. Alternatively, the client program generates this prompt responsive to locally stored information. Alternatively or additionally, the customer can make a special request to continue a previous session.

A fifth aspect of some embodiments of the present invention relates to associating a password with the session ID. When files are uploaded, such as using an FTP protocol, they are uploaded into a protected location, using the user and/or the password. Preferably, the password and/or a user name are selected from a bank of names, which are reused. If the bank is large enough, it is very difficult to break in to the system using an old password. Preferably, the service provider associates a particular customer with a user/password combination and does not allow connections to that user other than from the particular customer, for a certain limited period of time. Alternatively, each session is assigned a unique password and/or user name, preferably related to the unique session ID.

A sixth aspect of some embodiments of the present invention is related to a process structure preferred for preferred embodiments of the invention. Preferably, the service provider includes a WWW server, for the interactive session and an FTP server for the file upload session. Preferably, both servers are standard commercial software products, so that, preferably, they can be controlled externally, as black boxes. Preferably, each of the FTP server and the WWW server run on different machines. In a preferred embodiment of the invention, the service provider includes a synchronizing process which supplies the unique session ID and synchronizes the operation of the FTP server and the WWW server. Alternatively or additionally, the customer software may be provided with a synchronizing process. Alternatively or additionally, at least part of the synchronization is achieved from a third location. In a preferred embodiment of the invention, the customer software includes a standard WWW browser.

It should be appreciated that in some embodiments of the present invention only one or several of the above described aspects will be practiced.

There is therefore provided in accordance with a preferred embodiment of the invention, a method of synchronizing an interactive connection and a non-interactive data transfer connection between a client and a service provider, comprising:

- creating an interactive connection;
- creating a data transfer connection; and
- generating a session ID which is associated with the two connections.

There is also provided in accordance with a preferred embodiment of the invention, a method of transferring data between a client and a service provider, comprising:

- creating an interactive connection between the client and the service provider;
- creating a data transfer connection between the client and the server;

automatically uploading data files from the client to the server, responsive to the interactive connection.

Preferably, automatically uploading comprises automatically stopping uploading of data files which are indicated through the interactive connection as being unnecessary.

Alternatively or additionally, automatically uploading comprises automatically repeating uploading of data files which are indicated through the interactive connection as being necessary and which were modified at the client between being first uploaded and being second uploaded.

Alternatively or additionally, automatically uploading comprises automatically uploading only data files which are indicated through the interactive connection as being necessary.

Alternatively or additionally, the method comprises generating a single session ID for the two connections. Preferably, automatically uploading comprises associating said uploaded data files with said session ID. Alternatively or additionally, the method comprises providing, from the service provider, a user name and wherein automatically uploading data files comprises automatically uploading data files using the provided user name. Preferably, the method comprises:

- associating said user name with a particular client; and
- rejecting, for a limited period of time, connections to the user name, which are not from the particular client.

There is also provided in accordance with a preferred embodiment of the invention, apparatus for uploading data files, comprising:

- a file upload connection server;
- an interactive connection server; and
- a synchronizer which synchronizes the operation of the two connections.

Preferably, said synchronizer generates a single session ID for two associated sessions, each on a different one of said servers. Alternatively or additionally, said synchronizer receives from a client, through said interactive connection, a list of file locations and instructs said file upload connection client to upload said files to said server. Alternatively or additionally, said synchronizer and said interactive connection server are integrated as a single process.

Alternatively or additionally, said synchronizer and said file upload server are integrated as a single process.

Alternatively, said file upload server and said interactive connection server are each on a separate computer.

There is also provided in accordance with a preferred embodiment of the invention, apparatus for uploading data files, comprising:

- a file upload connection client;
- an interactive connection client; and
- a client synchronizer which synchronizes the operation of the two connections.

Preferably, said clients and said synchronizer are integrated in a single client computer.

Alternatively or additionally, the apparatus comprises:

- a file upload connection server; and
 - an interactive connection server,
- wherein said servers are on connected to said clients by computer communications. Preferably, the apparatus comprises a server synchronizer which synchronizes, together with the client synchronizer the operation of the two connections.

There is also provided in accordance with a preferred embodiment of the invention, apparatus for synchronizing a file upload connection and an interactive connection, comprising:

an file upload monitor, which monitors the operation of a file upload server without direct communication with the file upload server;

an interactive data generator, which generates data in a format suitable for an interactive connection server, and a synchronizer,

wherein said synchronizer causes said interactive data generator to generate data responsive to input from said file upload monitor and which sends the generated data through the interactive connections server.

Preferably, said synchronizer generates a username for use of said file upload connection. Preferably, said file upload server is connected to a file upload client and said username is transmitted to said file upload client, for connecting to said file upload server using said username.

Alternatively or additionally, said file upload server is a FTP server. Alternatively or additionally, said interactive connection server is a WWW server.

In a preferred embodiment of the invention, the apparatus comprises a second synchronizer, which communicates with the synchronizer and controls the uploading files to the file upload server responsive to the communications. Preferably, said second synchronizer communicates with said first synchronizer on a connection used for uploading the files.

There is also provided in accordance with a preferred embodiment of the invention, a method of local file information display, comprising:

uploading a list or file information for a plurality of local files to a remote server;

generating a data display at the remote server; and

locally displaying said data display, wherein said data display includes local data responsive to said local file information.

Preferably, said uploading a list comprises automatically uploading a list. Alternatively or additionally, said local data comprises local images corresponding to said local file information, wherein said images are not yet uploaded to said remote server.

There is also provided in accordance with a preferred embodiment of the invention, a method of synchronized file upload, from a upload client to an upload server, comprising: connecting from said client to said server;

receiving information at said client from said server; and uploading files from said client to said server, utilizing said information.

Preferably, said server comprises a file upload server and a software component, external to said file upload server, which monitors the operation of said file upload server and which transmits said information to said client. Alternatively or additionally, said information comprises a username and wherein said uploading files comprises uploading files using said username. Alternatively or additionally, said information comprises an indication of at least one file whose upload failed and wherein uploading files comprises repeating the upload of at least said one file.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be more clearly understood from the detailed description of the preferred embodiments with reference to the accompanying figures, in which:

FIG. 1 is a schematic block diagram of a file upload system, in accordance with a preferred embodiment of the invention;

FIG. 2 is a schematic block diagram of a file upload system, in accordance with a preferred embodiment of the

invention, showing a synchronizing process as part of a service provider;

FIG. 3 is a schematic block diagram of a file upload system, in accordance with a preferred embodiment of the invention, showing a synchronizing process as part of a client;

FIG. 4 is a schematic block diagram of a file upload system, in accordance with a preferred embodiment of the invention, showing a synchronizing process as part of both a client and a service provider; and

FIG. 5 is a schematic block diagram of a file upload system, in accordance with a preferred embodiment of the invention, showing a synchronizing process as part of a third party.

DETAIL DESCRIPTIONS OF PREFERRED EMBODIMENTS

FIG. 1 is a schematic block diagram of a file upload system, in accordance with a preferred embodiment of the invention. The system includes a client 10 and a service provider 16. At least two connections are operational between client 10 and service provider 16: a file upload connection and an interactive connection. Preferably, the file upload connection is an FTP connection, preferably between an FTP client 12 in client 10 and an FTP server 18 in service provider 16. Alternatively to an FTP connection, a direct file transfer protocol may be used, such as Kermit. Preferably, the interactive connection is a WWW connection between a WWW client 14 in client 10 and a WWW server 20 in service provider 16. In a preferred embodiment of the invention, FTP server 18 and WWW server 20 are synchronized using a synchronization connection 22.

In a preferred embodiment of the invention both the connections are on a single physical line. Alternatively, two lines may be provided, for example, the WWW connection being on a LAN which is connected to an Internet, while the file upload being on a dedicated modem.

In a preferred embodiment of the invention, service provider 16 comprises a plurality of computers and at least FTP server 18 and WWW server 20 are each run on a separate computer.

The operation of the system, in accordance with a preferred embodiment of the invention is easily explained with reference to a particular application. The reference application used herein is an image manipulation application, preferably associated with a digital camera or a scanner device.

In the preferred image manipulation application, the scanner and/or camera are provided with an image manipulation program for manipulating images generated by the camera/scanner. One option of the image manipulation program is connecting to an outside service provider for special services which cannot be performed at home, for example, creating photographic-type hard copies or printing images on plastic objects.

Preferably, when the customer selects an outside service provider, an interactive connection is made through the Internet. However, other types of public computer networks may be used instead. The interactive connection is made between WWW client 14, which is preferably a standard browser and WWW server 20 which is preferably a commercially available WWW server. The image manipulation program preferably also opens a file upload connection, preferably, also through the Internet. However, in an alternative preferred embodiment of the invention, the two

connections may be through two different networks. The two networks may share the same physical layer, or they may use different physical layers. Service provider 16 preferably provides a unique session ID to the combined FTP and WWW connection. In a preferred embodiment of the invention, the FTP connection is started only after such a unique ID is provided by WWW server 20.

In a preferred embodiment of the invention, synchronization connection 22 is used to synchronize the two servers. FIGS. 2,5 describe four possible configurations for synchronizing client 10 and service provider 16, which will be described in greater detail below. In the configuration of FIG. 2, a synchronization process 30 is incorporated in service provider 16. In the configuration of FIG. 3, a synchronization process 40 is incorporated in client 10. In the configuration of FIG. 4, a synchronization process 50 is incorporated in service provider 16 and a synchronization process 52 is incorporated in client 10. In the configuration of FIG. 5, at least part of the synchronization is provided by a third party synchronization process 60.

Many features of preferred embodiments of the invention, described below, are performed by the synchronization process, the location of the synchronization process will affect both the way in which these features are implemented and whether they are possible to implement. In particular, some security features are not possible or will have a lower utility in some of the configurations.

In a preferred embodiment of the invention, FTP client 14 and FTP server 18 are standard commercial FTP programs. Alternatively, a limited FTP client which can only upload images in response to a uniquely provided session ID, is provided with the client software. Alternatively, an open FTP client is used, which can be easily integrated into the software at client 10. An open FTP client is especially preferred when at least part of the synchronization is by a program at client 10. In such a case, the synchronization process is preferably integrated with the FTP client. Furthermore, the FTP client and the synchronization process preferably share a TCP/IP connection.

Preferably, FTP client 12 starts uploading active image files as soon as an FTP connection is established, which is typically only after a session ID is obtained. Preferably, the upload starts with an upload of a list of all the active files. Thereafter, files are uploaded from FTP client 12 to FTP server 18, responsive to file names on the list. Typically, FTP client 12 initiates the upload, since FTP server 18 may have limited control options. Alternatively, the list of active files is provided through the WWW connection. Active files are a set of files which are considered by the image manipulation software to be relevant for the required interaction with the external service provider. Some examples of definitions for active files include: all the files which were previewed in the current run of the image manipulation program, all files which are still open in the program, all files which were marked as "OK" by the user of the program and/or a predefined set of files.

In a preferred embodiment of the invention, the user can interact with the service provider even before any images are uploaded, by viewing and manipulating thumbnail images, full size images or image names. This information may be displayed locally by the WWW client, by selectively displaying files from the above provided list of file names and/or locations, as thumbnail files, full size images and/or file names. Alternatively or additionally, the thumbnail images may be generated locally by the image manipulation software and passed to the WWW client, by passing their

location to the WWW server. Alternatively, thumbnail images are also uploaded to the service provider, preferably before uploading the complete image files. These thumbnail images are preferably used by the service provider to give the user an idea of what the manipulated image will look like.

As can be appreciated, at some point in the process, the user will be required to indicate, to the service provider, which of the active images are to be serviced. In some cases, making no selection will automatically select all the images. When the user, through the interactive connection, selects certain images, the FTP connection preferably stops uploading any unselected files and from that point on uploads only the selected files.

In some cases, the user will finish ordering the service before the selected files are all uploaded. In such a case, service provider 16 will notify the user of his options, through the WWW connection. One option might be to cancel services related to images which were not uploaded. Another option might be to wait until all the images are uploaded. Yet another option might be to continue the upload in the background, while discontinuing the WWW connection. Still another option might be to continue the session later.

In a preferred embodiment of the invention, a user may stop a session and then return to the session at a later time utilizing the unique session ID to identify the session. Preferably, the existence of an unfinished session is identified either by client 10 or by service provider 16 and the user may be provided with an option to reconnect to an old session. In some cases a new unique session ID number may be provided even when an old session is continued.

In a preferred embodiment of the invention, the unique session ID is unique to a particular client. In one example, the unique session ID include a secret client code which is embedded in the client software. One example of such a code is the client's IP address, preferably appended to a time stamp.

In many cases, some of the upload attempts will fail and some files will fail to be uploaded. Preferably, the FTP connection will not retry the failed files until all the active files have been uploaded and/or until certain files are selected as participating in the service. Preferably, unselected files will not be reloaded.

In some cases, the user will manipulate images after they are uploaded. Preferably, the service provider will automatically upload the modified images. Alternatively, the service provider will prompt the user, through the WWW connection, of his options, which may include using the old file, using the new file or canceling the file. Alternatively, files to be uploaded are frozen, such as by copying the files to a special upload directory.

Referring to FIG. 2, in a preferred embodiment of the invention, synchronization between the interactive connection and the file transfer connection is achieved using synchronization process 30 in service provider 16. In this embodiment, the synchronization information is preferably carried on the same TCP/IP connection which is used for the file upload, albeit, preferably using a different port. Preferably, synchronizing process 30 monitors connections to FTP server 18. In this configuration, the unique session ID is preferably generated by synchronization process 30.

In a preferred embodiment of the invention, file upload is monitored indirectly, either by communication between FTP client 12 and synchronizing process 30 or by using uploaded files as semaphores. Synchronizing process 30 can deter-

mine whether a file was uploaded successfully by checking if a corresponding upload-OK file was uploaded.

In a preferred embodiment of the invention, FTP client 12 is a stand-alone commercial program. The file upload is preferably started by WWW browser 14 spawning the FTP application. Preferably, the spawn command is generated by synchronizing process 30 and passed through WWW server 20 to WWW browser 14. Preferably, a separate spawn command is generated for each file to be uploaded. Alternatively or additionally, the spawn command includes unique session ID information and/or user/password information (described below). Alternatively or additionally, the file upload is achieved by downloading a program, preferably a batch file, to upload the files. It should be appreciated that in many WWW browsers, spawning executables is restricted for security reasons. It should also be appreciated that the FTP program is usually located in a standard location for a particular operating system. Additionally or alternatively, the image manipulation program uploads the location of the FTP command. Additionally or alternatively, the FTP program is included in the image manipulation software package. Thus, it is possible, in some embodiments of the present invention, to only allow executing a particular program from WWW browser 14.

In a preferred embodiment of the invention, the step of receiving a unique session ID includes receiving a temporary user name and/or a password. Preferably, FTP client 12 uploads the image files, preferably to a unique location, using the provided user name and/or password. Additionally or alternatively, each uploaded file is associated with a unique session ID, preferably, by storing the file in a directory associated with the session ID or by appending the session ID to the name. In one preferred embodiment of the invention, the unique session ID is used to generate a unique user name and/or password. Alternatively, a bank of available user names may be used, from which a user name is cyclically selected. Thus, a user name will generally not be used simultaneously by two different customers. In some cases, especially when there are more active connections than user names, two customers will share a single user name. However, this should pose no problem, since files are still individually identified using their associated session IDs. Alternatively or additionally, the user name and/or the password are encrypted in the session ID, preferably using a public key encryption scheme. Thus, unique identification of the owner of the connection can be assured.

In a preferred embodiment of the invention, the passwords are periodically changed, such as every day.

User names assigned to a previous session are preferably frozen until the session is terminated. Alternatively, data files associated with a session are removed from the particular user account and are moved to a new user which is provided with the next connection.

In a preferred embodiment of the invention, synchronizing process 30 vets incoming FTP connections and limits connections to a particular user name to those connections arriving from a particular client. The client is preferably identified by its name, a user supplied self identification such as a software code and/or the location of the client. Thus, connecting to a particular user account is farther protected from attempts to break into the system. Alternatively, synchronizing process 30 can disconnect suspicious connections. Alternatively or additionally, this technique is used to conserve user names, by using the same user name for all connections from a particular local. Alternatively or additionally, this technique is used to stop uploads of files

which are not necessary. In a preferred embodiment of the invention, client 10 uploads all the files and synchronizing process 30 stops the upload of unnecessary files, such as by disconnecting the FTP connection.

Referring to FIG. 3, in a preferred embodiment of the invention, synchronization between the interactive connection and the file transfer connection is achieved using synchronizing process 40 in client 10. Preferably, synchronizing process 40 and FTP client 12 are integrated. Preferably, FTP client 12 is conducted using open standard code, which is generally available as compiled libraries. In this embodiment, preferably no synchronization connection is necessary. Alternatively, synchronization information is preferably carried on the same TCP/IP connection which is used for the file upload. Preferably, such information is provided by file down load.

In a preferred embodiment of the invention client 10 provides a unique session ID, for example by providing a unique user identification and a local session number. This is especially relevant where a client will reuse the same service several times and therefore obtains a unique user identification from the service. In some cases, client 10 will have a pre-provided user name to which directory the files will be uploaded. However, in most cases, client 10 will be an infrequent or one time user of the provided service and will therefore not be assigned a unique user name in advance. Alternatively, client 10 provides a unique session ID, such as based on its IP number and/or a time stamp. It should be noted that in this embodiment, usually no user name will be assigned by service provider 16.

Referring to FIG. 4, in a preferred embodiment of the invention, synchronization between the interactive connection and the file transfer connection is achieved using synchronizing process 50 in service provider 16 and synchronizing process 52 in client 10. In this embodiment, the synchronization between the two synchronizing processes is preferably carried on the same TCP/IP connection which is used for the file upload. Alternatively, a separate TCP/IP control connection may be provided. In a preferred embodiment of the invention, the session ID is provided by client 10, while the user name is provided by service provider 16. As described above for other embodiments, synchronizing process 52 is preferably integrated with FTP client 12 and preferably generates the unique session ID. With reference to FIG. 2, it should be appreciated that in preferred embodiments of the invention, the image manipulation software does perform some synchronization, such as generating a list of active images to be uploaded and/or generating a unique session ID. However, this software does not generally coordinate the file upload and the interactive session once they have started.

Referring to FIG. 5, in a preferred embodiment of the invention, synchronization between the interactive connection and the file transfer connection is achieved using synchronizing process 60 external to service provider 16 and client 10.

Synchronization process 60 may be in addition to or instead of the synchronizing processes in the embodiments of FIGS. 2-4, although, preferably, there is also a synchronizing process in service provider 16. In one example, synchronizer 60 is located at a second service provider and WWW server 20 is either a proxy to a second WWW server at the second provider or located at the second service provider. Files are uploaded to service provider 16 and synchronizer 60 detects their upload and performs a service responsive to the upload and instructions from WWW server

20. Proper uploading of files may be detected using semaphore files, as described above. It should be noted that this monitoring of the uploading may be performed remotely, such as through an FTP connection. Alternatively or additionally, service provider 16 further uploads the files to the second service provider. Synchronizer 60 will preferably provide a user name to be passed or directly transmitted to client 10. In another example, it is the FTP server which is at the second service provider, while synchronizing process 60 is preferably at service provider 16. Thus, files are upload directly to the second service provider, while the interactive session is handled at service provider 16.

Although preferred embodiments of the present invention have been described mostly with reference to a WWW connection and an FTP connection, other types of connection protocols may be used. Further, more than two connections may be current, for example, more than one interactive connection and/or more than one file upload connection.

It should be noted that although many of the above embodiments require both an interactive session and a file upload session, some aspects of the invention do not require an interactive session. In particular, password verification for file upload does not require an interactive session, nor does automatic reloading of files which were not successfully uploaded.

In a preferred embodiment of the invention, an FTP file upload server and an FTP file upload client each include a synchronizing portion. The FTP client synchronizer portion contacts the FTP server synchronizer portion and requests a password. The files are uploaded using the password and/or a unique session ID which may be generated at the client or at the server, as described above. If a file is not successfully uploaded, the FTP server synchronizer portion can detect this, as described above, and instruct the FTP client synchronizer portion to upload the file again. Thus, secret and guaranteed uploading of files, using a public network can be achieved, with minimal required changes in an FTP client and with no changes in an FTP server, except the addition of external programs. This embodiment is especially useful, where it is desirable for the images to be uploaded prior to contacting a service provider and the image upload time is long. Automatic upload of images may require user instructions. Alternatively, active images and/or images which have been processed in a certain way are automatically uploaded unless otherwise instructed by the user.

Attached herewith is a software appendix of a system for synchronized file upload using an interactive connection, in accordance with a preferred embodiment of the invention. The software system is not completely debugged, however, it will operate, in most cases, in a suitable fashion. The file upload client the one provided with Microsoft visual C/C++ version 5.0, the programming language in which the client is programmed. The server is programmed in Java. It will be appreciated by a person skilled in the art that the present invention is not limited by what has thus far been described. Rather, the present invention is limited only by the claims which follow.

We claim:

1. A method of synchronizing an interactive connection and a non-interactive data transfer connection between a client and a service provider, comprising:
creating an interactive connection;
creating a data transfer connection; and
generating a single session ID for the two connections, which ID associates between the two connections.
2. A method according to claim 1, wherein the data transfer connection is created responsive to the interactive connection.

3. A method according to claim 1, wherein the data transfer connection and the interactive connection are created concurrently.

4. A method according to claim 1, wherein creating the data transfer connection comprises creating a connection substantially only for uploading data files from the client to the service provider.

5. A method according to claim 1, wherein creating the data transfer connection comprises creating a connection which is used for uploading files from the client to the service provider and wherein creating the interactive connection comprises creating a connection which transmits information relating to the uploaded files.

6. A method according to claim 5, wherein the interactive connection transmits indications of the uploaded files.

7. A method according to claim 1, wherein creating the data transfer connection comprises creating a connection which is used for uploading data files from the client to the service provider and wherein creating the interactive connection comprises creating a connection which transmits information relating to the upload of the files.

8. A method according to claim 7, comprising modifying the upload of data files responsive to at least one command on the interactive connection.

9. A method according to claim 7, wherein modifying the upload of data files comprises modifying the order of upload of the files.

10. A method according to claim 7, wherein modifying the upload of data files comprises canceling the upload of one or more of the files.

11. A method of transferring data between a client and a service provider, comprising:

creating an interactive connection between the client and the service provider;

creating a data transfer connection between the client and the service provider; and

automatically uploading data files from the client to the service provider, on the data transfer connection, responsive to the interactive connection.

12. A method according to claim 11, wherein automatically uploading comprises automatically stopping uploading of data files which are indicated through the interactive connection as being unnecessary.

13. A method according to claim 11, wherein automatically uploading comprises automatically repeating uploading of data files which are indicated through the interactive connection as being necessary and which were modified at the client between being first uploaded and being second uploaded.

14. A method according to claim 11, wherein automatically uploading comprises automatically uploading only data files which are indicated through the interactive connection as being necessary.

15. A method according to claim 11, comprising generating a single session ID for the two connections.

16. A method according to claim 15, wherein automatically uploading comprises associating said uploaded data files with said session ID.

17. A method according to claim 15, comprising providing, from the service provider, a user name and wherein automatically uploading data files comprises automatically uploading data files using the provided user name.

18. A method according to claim 17, comprising:
associating said user name with a particular client; and
rejecting, for a limited period of time, connections to the user name, which are not from the particular client.

19. A method according to claim 11, wherein creating the data transfer connection comprises creating the connection responsive to the creation of the interactive connection.

20. A method according to claim 11, wherein automatically uploading data files comprises automatically determining an order of upload of the data files.

21. A method according to claim 11, wherein automatically uploading comprises automatically determining which files are uploaded.

22. A method according to claim 11, wherein automatically uploading data files comprises uploading responsive to previewing of the data files.

23. A method according to claim 11, wherein automatically uploading data files comprises uploading responsive to modifying the data files.

24. A method according to claim 11, wherein automatically determining which files are uploaded comprises deciding by a computer which files are uploaded.

25. Apparatus for uploading data files, comprising:

a file upload connection server;

an interactive connection server; and

a synchronizer which synchronizes the operation of respective connections formed by the file upload connection server and by the interactive connection server.

26. Apparatus according to claim 25, wherein said synchronizer generates a single session ID for two associated sessions, each on a different one of said servers.

27. Apparatus according to claim 25, wherein said synchronizer receives from a client, through said interactive connection, a list of file locations and instructs said file upload connection client to upload said files to said server.

28. Apparatus according to claim 25, wherein said synchronizer and said interactive connection server are integrated as a single process.

29. Apparatus according to claim 25, wherein said synchronizer and said file upload server are integrated as a single process.

30. Apparatus according to claim 25, wherein said file upload server and said interactive connection server are each on a separate computer.

31. Apparatus according to claim 25, wherein the synchronizer monitors connections to the file upload connection server.

32. Apparatus for uploading data files, comprising:

a file upload connection client;

an interactive connection client; and

a client synchronizer which synchronizes the operation of respective connections formed by the file upload connection client and by the interactive connection client.

33. Apparatus according to claim 32, wherein said clients and said synchronizer are integrated in a single client computer.

34. Apparatus according to claim 32, comprising:

a file upload connection server; and

an interactive connection server,

wherein said servers are connected to said clients by computer communications.

35. Apparatus according to claim 34, comprising a server synchronizer which synchronizes, together with the client synchronizer the operation of the two connections.

36. Apparatus for synchronizing a file upload connection and an interactive connection, comprising:

an file upload monitor, which monitors the operation of a file upload server without direct communication with the file upload server;

an interactive data generator, which generates data in a format suitable for an interactive connection server; and

a synchronizer,

wherein said synchronizer causes said interactive data generator to generate data responsive to input from said file upload monitor and which sends the generated data through the interactive connections server.

37. Apparatus according to claim 36, wherein said synchronizer generates a username for use of said file upload connection.

38. Apparatus according to claim 37, wherein said file upload server is connected to a file upload client and wherein said username is transmitted to said file upload client, for connecting to said file upload server using said username.

39. Apparatus according to claim 36, wherein said file upload server is a FTP server.

40. Apparatus according to claim 36, wherein said interactive connection server is a WWW server.

41. Apparatus according to claim 36, comprising a second synchronizer, which communicates with the synchronizer and controls the uploading files to the file upload server responsive to the communications.

42. Apparatus according to claim 41, wherein said second synchronizer communicates with said first synchronizer on a connection used for uploading the files.

43. A method of local file information display, comprising:

uploading a list of file information for a plurality of local files to a remote server;

generating a data display at the remote server; and

locally displaying said data display, wherein said data display includes local data not downloaded from the remote server, responsive to said local file information.

44. A method according to claim 43, wherein said uploading a list comprises automatically uploading a list.

45. A method according to claim 43, wherein said local data comprises local images corresponding to said local file information, wherein said images are not yet uploaded to said remote server.

46. A method of synchronized file upload, from an upload client to an upload server, comprising:

connecting from said client to said server;

receiving information comprising a username at said client from said server; and

uploading files from said client to said server, utilizing said information.

47. A method according to claim 36, wherein said server comprises a file upload server and a software component, external to said file upload server, which monitors the operation of said file upload server and which transmits said information to said client.

48. A method according to claim 36, wherein said information comprises an indication of at least one file whose upload failed and wherein uploading files comprises repeating the upload of at least said one file.

* * * * *



US005751719A

United States Patent [19]

Chen et al.

[11] **Patent Number:** 5,751,719[45] **Date of Patent:** May 12, 1998[54] **METHOD AND SYSTEM FOR DATA TRANSFER IN THE PRESENCE OF DISCONNECTS**[75] **Inventors:** Kuo-Wei Herman Chen, Scotch Plains; Sanjoy Paul, Atlantic Highlands; Krishan Kumar Sabnani, Westfield, all of N.J.[73] **Assignee:** Lucent Technologies Inc., Murray Hill, N.J.[21] **Appl. No.:** 565,526[22] **Filed:** Nov. 30, 1995[51] **Int. Cl.⁶** H04L 1/16; G08C 25/00[52] **U.S. Cl.** 370/473; 370/394; 371/32; 395/200.12[58] **Field of Search** 370/349, 469, 370/471, 473, 474, 394; 395/200.12, 200.13, 200.14, 182.16, 182.18, 181; 371/32, 34[56] **References Cited****U.S. PATENT DOCUMENTS**

4,617,657	10/1986	Drynan et al.	370/394
4,841,526	6/1989	Wilson et al.	371/32
5,151,899	9/1992	Thomas et al.	370/394
5,222,061	6/1993	Doshi et al.	370/394
5,245,616	9/1993	Olson	371/32
5,444,718	8/1995	Ejzak et al.	371/32

OTHER PUBLICATIONS

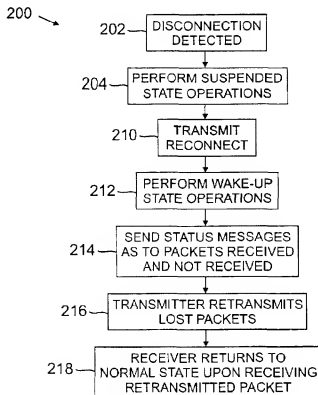
A.S. Tanenbaum, Computer Networks, pp.10-21, Prentice Hall Software Series, 1981.

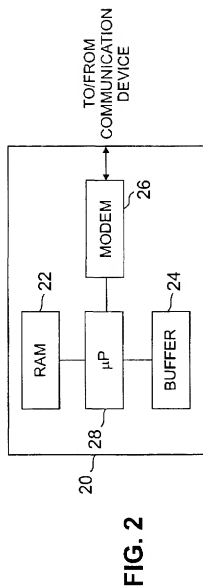
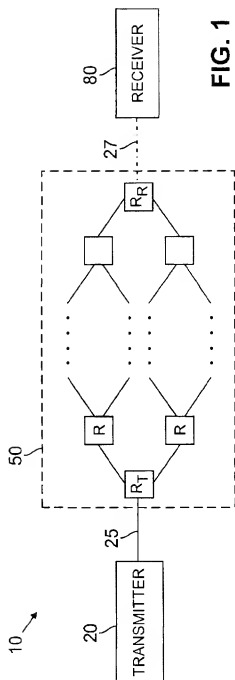
E. Ayanoglu, S. Paul, T.F. La Porta, K.K. Sabnani, R.D. Gitlin, "AIRMAIL: A Link-layer Protocol for Wireless Networks," Wireless Networks vol. 1, No. 1, pp. 47-60, 1995.

S. Aggarwal, K. Sabnani, B. Gopinath, A New File Transfer Protocol, AT&T Technical Journal, vol. 64, No. 10, pp. 2387-2411, Dec. 1985.

Primary Examiner—Hassan Kizou[57] **ABSTRACT**

Methods and systems for controlling data transfer operations during a communication session between a transmitter and at least one receiver provide that data packets which have been stored at the receiver memory before a disconnect occurred remain stored therein in the presence of the disconnect to avoid the need for retransmitting data packets upon re-establishment of a communication link. Data representative of those data packets already stored at the receiver memory is maintained for purposes of synchronizing data packet transmission from the transmitter for completing data file transfer. Data control instructions for implementing storage of data packets and tracking the progression of data packet transmissions when a disconnect occurs may suitably be included in an existing protocol layer of a communication system or as an additional protocol layer added to the operating system of the communication system.

18 Claims, 8 Drawing Sheets



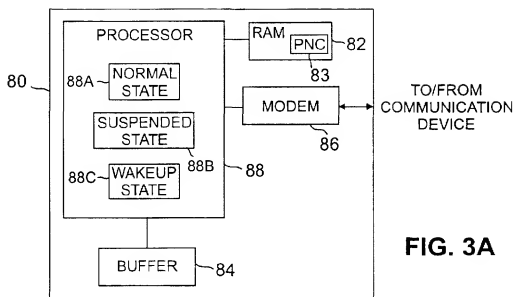


FIG. 3A

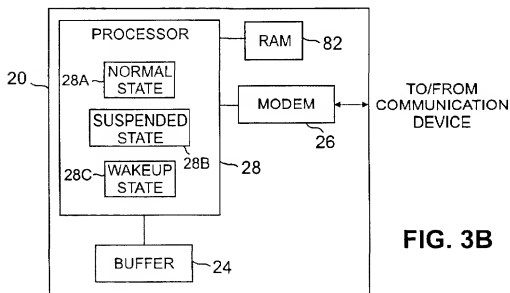


FIG. 3B

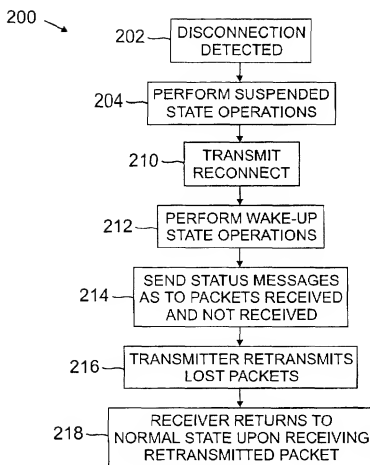


FIG. 3C

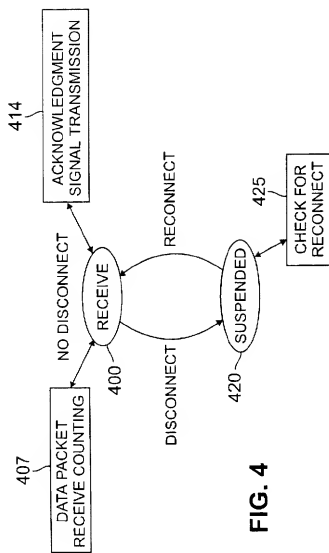


FIG. 4

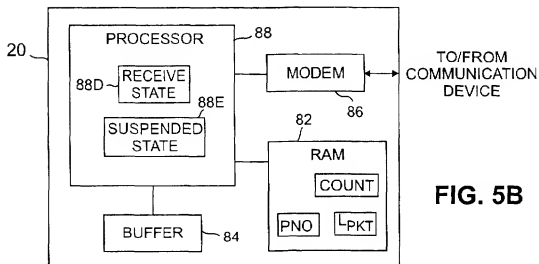
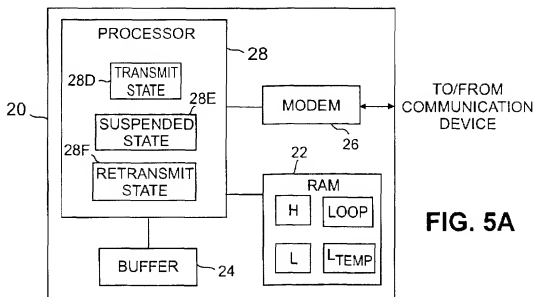


FIG. 6A

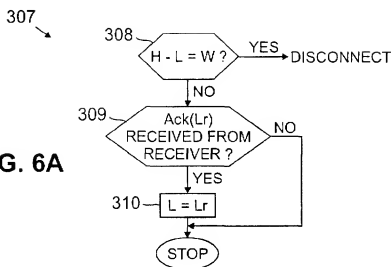
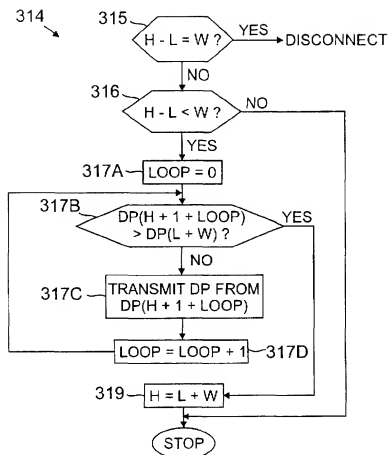
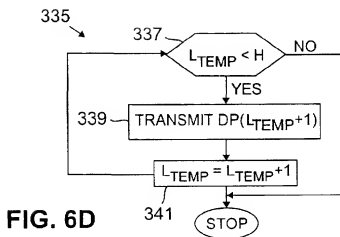
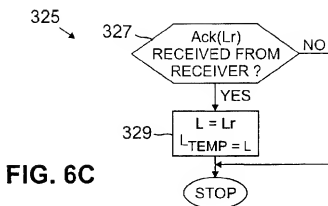


FIG. 6B





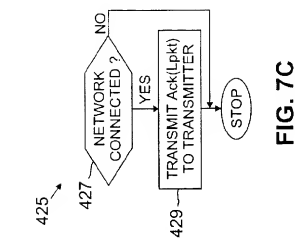


FIG. 7C

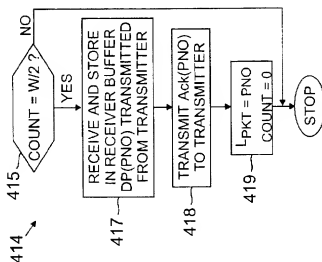


FIG. 7B

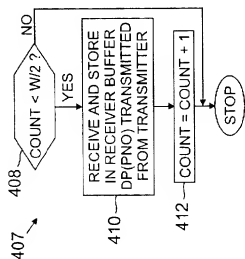


FIG. 7A

METHOD AND SYSTEM FOR DATA TRANSFER IN THE PRESENCE OF DISCONNECTS

FIELD OF THE INVENTION

This invention relates generally to communication systems which provide for the transmission of sequentially numbered data packets from a transmitter to a receiver. More particularly, the present invention concerns techniques for controlling data transfer operations for minimizing data retransmissions when a disconnect occurs during the transmission of sequentially numbered data packets from a transmitter to a receiver.

BACKGROUND OF THE INVENTION

Data communication systems which provide for the transmission of a data file using sequentially numbered data packets are known. These systems conventionally operate according to standard communication system control procedures known as protocols. The protocols define the system operating parameters associated with transferring data over a communication link from a transmitter to a receiver during a communication session.

A standard communication operating system is typically arranged as a hierarchy of protocol layers for controlling data handling operations during data transfer. For example, a point-to-point signal transfer across physical media, such as a cable, is controlled by protocol layers known as physical or data link protocol layers, while signal transfer on an end-to-end basis, in other words, between a transmitting and receiving communication device, is controlled at protocol layers known as transport or application protocol layers. [See A. S. Tannenbaum, *Computer Networks*, pp. 20-21, 1981 citation, to be provided by inventor], incorporated by referenced herein, for a detailed description of the operations performed at different protocol layers in a communication system.

In a communication system, the protocols establish a virtual connection between a transmitting communication device and at least one receiving device for transferring data from the former to the latter during a communication session. For example, the protocols may provide that a personal computer may transfer data packets of a data file through a wired interconnecting network and ultimately to a wireless mobile unit, such as, for example, a portable laptop computer or personal digital assistant.

During the course of a communication session, a communication link between a transmitting and a receiving device may become disconnected due to breakdowns in data links or the network, or poor signal transmission between the former and latter. These disconnects are known to occur very frequently in a mobile networking environment. For example, a wireless radio channel may temporarily become nonfunctional or the battery in a mobile receiver may become too weak, causing a disconnect to occur.

When a disconnect occurred in a prior art communication system, a protocol associated with receiving, storing and processing data received during a communication session, such as, for example, the data transfer application protocol associated with reconstructing images or text from a data file, often became destabilized. The destabilization of such a protocol layer interrupted the normal processing operations of the protocol and, in some circumstances, caused the loss of any data previously received and stored in the memory of a receiving device.

The loss of stored data which has been received as part of a data file transfer is especially disadvantageous where a

protocol requires that an entire data file be stored before the data file may be processed. For example, a data transfer application protocol layer associated with reconstructing images from data may require an entire data file to be received and stored in memory before the data file is processed for the re-creation of images. The necessity for retransmitting an entire data file because of a disconnect, thus, severely degrades throughput and becomes especially burdensome where large amounts of data must be retransmitted.

SUMMARY OF THE INVENTION

The present invention is directed to methods and systems for controlling data file transfer operations in a communication system during a disconnect to avoid unnecessary retransmissions of data packets once a communication link is re-established.

In one aspect of the present invention, an existing protocol of a communication operating system is modified to include programmed instructions, or a recovery module, for controlling data handling operations in the event a disconnect occurs during data file transfer. During a communication session in the system, a data file is transmitted from a transmitting device to a receiving device as a numbered sequence of data packets. When a disconnect is detected, the recovery module provides that data packets received and stored at the receiving device prior to the occurrence of the disconnect are retained in memory therein. The data packets are stored in the memory to permit that their respective sequence numbers may be obtained subsequently. After a connection is re-established, the sequence numbers of the data packets which have been previously received and retained in the memory of the receiving device during the disconnect are utilized for indicating to the transmitting device which data packets still need to be transmitted in order to complete transmission of the data file.

In another aspect of the invention, a separate and independent protocol layer, or recovery layer, is included in a communication operating system hierarchy above an existing protocol layer which may become destabilized in the event of a disconnect. The recovery layer provides, during a communication session between a transmitting and receiving device, that the receiving device maintain a record of which data packets have been received and stored. The data packets are suitably transmitted from the transmitting device using a checkpointing window for facilitating tracking and monitoring of data packets received and stored at the receiving device. Once a connection is re-established after a disconnect occurs, the transmitting device is directed by the receiving device to transmit only those data packets required for completing the data file transfer. Data packet transmission is synchronized for checkpointing window transmission based on those data packets already stored in the memory of the receiving device.

Further features and advantages of the present invention will become readily apparent to those of ordinary skill in the art by reference to the following detailed description and accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows in block diagram form an exemplary communication switching system which operates in accordance with the present invention for controlling data transfer operations in the event of a disconnect during a communication session.

FIG. 2 illustrates an exemplary communication device in the system of FIG. 1 including functional blocks which may

3

suitably perform operations associated with the present inventive technique of controlling data transfer operations in the event of a disconnect during a communication session.

FIG. 3A illustrates an exemplary receiving device including functional blocks for implementing the present inventive technique of using a recovery module in an existing protocol layer of a communication system for handling disconnects.

FIG. 3B illustrates an exemplary transmitting device including functional blocks for implementing the present inventive technique of using a recovery module in an existing protocol layer of a communication system for handling disconnects.

FIG. 3C is a flow diagram of a method for controlling data transfer when a disconnect occurs in accordance with programmed instructions which are implemented as a recovery module in an existing protocol layer in accordance with the present invention.

FIG. 4 is a state diagram illustrating process states, transitions and action routines for a receiver operating in accordance with programmed instructions included in a recovery protocol layer of a communication system in accordance with the present invention.

FIG. 5A illustrates an exemplary transmitting device which may perform operations for controlling data transfer when a disconnect occurs in accordance with programmed instructions which are implemented as a separate recovery protocol layer in a communication protocol system in accordance with the present invention.

FIG. 5B illustrates an exemplary receiving device which may perform operations for controlling data transfer when a disconnect occurs in accordance with programmed instructions which are implemented as a separate recovery protocol layer in a communication protocol system in accordance with the present invention.

FIGS. 6A, 6B, 6C and 6D are flow diagrams of methods which may be performed by a transmitting device operating in accordance with the present invention.

FIGS. 7A, 7B and 7C are flow diagrams of methods which may be performed by a receiving device operating in accordance with the state diagram of FIG. 4.

DETAILED DESCRIPTION

FIG. 1 shows an exemplary communication system 10 which may suitably be operated in accordance with a protocol that controls data transfer operations for avoiding the retransmission of an entire data file of data packets when a disconnect occurs during a communication session, by enabling data transfer application processing to continue uninterrupted.

Referring to FIG. 1, the system 10 may suitably comprise a communication device 20 which is connected to a communication device 80 via an interconnecting network 50. The network 50 suitably comprises a plurality of routers or switching systems, R, which are interconnected to each other via land-line data links in accordance with standard techniques. The routers R are standard, well known components that may receive data signals from a communication device or router and then transmit these data signals to another router or communication device. It is to be understood that the network 50 may include wireless data links between the routers R, or there may be a combination of land-line and wireless data links which interconnect the routers R in the network 50. Further, it is to be understood that the system 10 may include additional communication devices which are connected to the network 50 using well known techniques.

4

FIG. 2 shows an exemplary embodiment of the device 20 comprising individual functional blocks. The functions these blocks represent may be provided through the use of either shared or dedicated hardware, including but not limited to hardware capable of executing software. In a preferred embodiment, the functional blocks in the device 20 comprise a random access memory (RAM) 22, a buffer memory 24 and a modem 26, all of which are connected to a standard microcontroller or microprocessor 28. As explained below, the functional blocks provide for the processing of the contents of message signals received at the device 20 and the processing of data associated with implementation of the present inventive technique. The processor 28 conventionally stores processed data results and data values received in a message signal in the RAM 22, and stores data packet information in the buffer 24. It is to be understood that the operations performed by the functional blocks and the processor 28 may be similarly performed using a single shared processor. Such a processor may comprise a standard digital signal processor and would include read only memory or other appropriate memory for storing software and performing the operations discussed below. Further, the present invention technique may be implemented using firmware or by discrete components implemented in an integrated circuit. Other suitable embodiments may be readily implemented by those of ordinary skill in the art.

Referring to FIG. 2, the modem 26 is suitably a standard component for receiving and transmitting message signals over either land lines or wireless data links. Further, the modem 26 suitably includes a processor and memory, not shown, for evaluating data packet transmission quality and invoking an interrupt, in other words, causing a disconnect in a communication link, when data transmission quality falls below a predetermined level. The buffer 24 is a conventional memory store for data packets which are either received or to be transferred during data file transfer. The RAM 22 is used for storing programmed instructions associated with the communication system protocols which control data file transmission operations. The processor 28 performs these instructions according to well known techniques. It is to be understood that additional communication devices may suitably be employed in the system 10 and that such devices will preferably, be structurally similar to and operate in a similar manner as the device 20. Operation will occur in accordance with the communication protocol. For example, the device 80 suitably includes a RAM 82, a modem 86 and a buffer memory 84, each of which is connected to a processor 88. A communication device may suitably be any electronic device, such as, for example, a personal or laptop computer or a personal digital assistant which includes electronic hardware, such as a modem, for transmitting and receiving electronic data signals message signals over a communication network.

For purposes of illustration, the present invention is explained with reference to operations performed in the system 10 when a disconnect occurs during a communication session involving data transfer from the device 20 to the device 80. For ease of reference, the devices 20 and 80 are hereinafter referred to as the transmitter 20 and the receiver 80, respectively. It is assumed that a virtual connection through the network 50 involves connection of the transmitter 20 via a land-line data link 25 to a router R_T and connection of the receiver 80 to a router R_R via a wireless data link 27. For the communication session involving a data file transfer from the transmitter 20 to the receiver 80, the router R_T is suitably connected through data links and routers in the network 50 for establishing a virtual commu-

5

nication link between the router R_R to the router R_T , in effect, between the transmitter 20 and the receiver 80.

As more fully described in Tannenbaum, cited above, data transfer operations in a communication system are controlled by the hierarchical protocol layers of the communication operating system. A data file transferred during a communications session may contain data packets representative of a text or an image. The data file includes a plurality of sequentially numbered data packets which are transmitted sequentially during a communication session. For example, a data file may include one hundred data packets numbered DP 00, DP 01 . . . DP 99 which are transmitted sequentially starting from DP 00. The data packets are suitably transmitted as message signals, wherein each message signal includes, in addition to the information contained in the data packet, data transfer processing data packets and control header information, such as the sequence number of the data packets as is conventional in the art.

For purposes of illustration, it is assumed that the communication operating system of the communication system 10 includes conventional end-to-end protocol layers called application and transport protocol layers, a network protocol layer concerned with the routing of signals within the network 50, and data link and physical media protocol layers concerned with point-to-point signal transmission, such as, for example, data signal transmission between the router R_T and the receiver 80 via the link 27. In the exemplary system 10, the hierarchy of protocol layers, in descending order, includes an application protocol layer, a transport protocol layer, a network protocol layer and a data link protocol layer. It is to be understood that the system 10 may include additional protocol layers and that those layers referred to herein are described only for purposes of illustration.

In accordance with the present invention, a communication system operating protocol provides a communication system with the capability of continuing to execute data transfer application processing in the event of a disconnect for avoiding the need for retransmitting data packets which were previously correctly received and stored during the communication session before the disconnect occurred. The inventive protocol provides that the correctly received data packets remain stored in a buffer in the event of a disconnect, and that the sequence numbers of those data packets correctly received and stored are tracked for enabling that only those data packets of a data file not previously received and stored are retransmitted, or originally transmitted, once a communication link is restored. The need for retransmitting data packets which have already been correctly received and stored at a device is avoided because correctly received data packets remain stored in memory in the event of a disconnect.

In one aspect of the present invention, a recovery module comprising programmed instructions is included in a protocol layer of a communication system which may become destabilized in the presence of a disconnect. A destabilized layer causes data packets correctly received and stored at a device to be eliminated from buffer memory. In a suitable embodiment, the recovery module is included within the transport protocol layer of the system 10 for stabilizing the transport protocol layer in the event of a disconnect. It is to be understood, however, that the recovery module may be included in any protocol layer in a communication system which needs to be stabilized in the event of a disconnect. A protocol layer in need of stabilization may include, for example, any end-to-end protocol layer of a communication operating system. The recovery module may be suitably included in the existing protocol with minor modifications

6

according to well known technique for interfacing modules in a protocol. For purposes of highlighting the advantages of the present invention, it is assumed that the transmitter 20 is sequentially transmitting a data file containing data packets numbered DP 00, DP 01 . . . DP 99.

FIG. 3A is an illustration of exemplary receiver 80 which may be utilized for implementing the present inventive technique of including the recovery module in an existing protocol layer. Referring to FIG. 3A, the receiver 80 may suitably comprise the interconnection of the RAM 82, the modem 86 and the buffer 84, as described above with reference to FIG. 2. The processor 88 comprises the functional processing blocks of NORMAL state processing component 88A, SUSPENDED state processing component 88B and WAKE-UP state processing component 88C for performing the operations described below with reference to FIG. 3C. A memory location 83 in the RAM 82 of the receiver 80 called a packet number counter (PNC), is utilized for storing data representative of the value of the sequence number of the last data packet which has been correctly received and stored in the buffer 84. The value of PNC indicates the data packets that still need to be transmitted for completing the transfer of a data file.

FIG. 3B is an illustration of exemplary transmitter 20 which may be utilized for implementing the present inventive technique of including the recovery module in an existing protocol layer. Referring to FIG. 3A, the transmitter 20 may suitably comprise the interconnection of the RAM 22, the modem 26 and the buffer 24, as described above with reference to FIG. 2. The processor 28 comprises the functional processing blocks of NORMAL state processing component 28A, SUSPENDED state processing component 28B and WAKE-UP state processing component 28C for performing the operations described below with reference to FIG. 3C.

FIG. 3C shows an exemplary method 200, which may be implemented as part of the recovery module, for controlling data transfer operations in the event of a disconnect to avoid retransmissions of data packets which already have been received and stored during a communication session. For implementation of the method 200, the system 10 is said to operate in a NORMAL state when data transfer is occurring in the absence of disconnects. In the NORMAL state, the NORMAL state processing component 88A performs programmed instructions associated with the transport protocol layer for storing in the buffer 84 data packets which have been correctly received at the modem 86 during a communication session. In the event the modem 86 or a router in the network 50 invokes a disconnect during a communication session, the recovery module of the transport protocol layer assumes control of data transfer operations, and the SUSPENDED state processing components 28B and 88B and the WAKE-UP state processing components 28C and 88C perform programmed instructions for stabilizing the transport protocol layer until a communication link is restored, such that transmission of data packets begins from the sequentially numbered data packet following the last data packet correctly received and stored at the buffer 84 of the receiver 80 before the disconnect occurred.

For purposes of highlighting the present inventive techniques, it is assumed that the modem 86 invokes a disconnect during the course of a communication session, and the disconnect occurs before all data packets of a data file have been received and saved at the buffer 84 of the receiver 80. The modem 86 suitably invokes an interrupt, in other words, causes a disconnect, after determining that data packets are being received at an error rate which exceeds a

predetermined threshold value. The modem 86, for example, may compute the signal-to-noise ratio (SNR) of the data packet transmission for a specified interval and compare the result with a predetermined SNR value for purposes of determining whether to invoke a disconnect. When the modem 86 invokes a disconnect, the physical connection between the wireless data link 27 and the receiver 80 is effectively eliminated, thereby severing the communication link between the transmitter 20 and the receiver 80.

The NORMAL state processing component 88A operates to store data at PNC in the RAM 82 for tracking the progression of data packet transmission. The value of PNC is updated for each data packet correctly received and stored at the buffer 84. For ease of reference, it is to be understood that reference hereinafter to the transmission and reception of message signals by a communications device, such as the reference 20, is a shorthand reference to the performance of such operations by the modem in the communications device in accordance with control signals suitably provided by the processing means of the device.

Referring to FIG. 3C, when a disconnect occurs, in step 202, the modem 86 provides control signals to the NORMAL state processing component 88A indicating the presence of a disconnect condition. The NORMAL state processing component 88A then ceases to perform NORMAL state operations associated with the transport protocol layer, and immediately thereafter, the SUSPENDED state processing component 88B begins to perform operations associated with the recovery module. Similarly, the SUSPENDED state processing component 28B begins to perform operations associated with the recovery module.

In step 204, the SUSPENDED state processing component 88B directs the modem 86 to attempt continuously to restore a communication link between the transmitter 80 and the receiver 20. For example, the modem 86 may be directed to dial the modem 22 in the transmitter 20 until a link is re-established. Further, the SUSPENDED state processing component 88B provides that those data packets correctly received and stored in the buffer 84 before the disconnect occurred continue to be retained therein. In prior art communication systems, data packets stored in the buffer of a communications device were not always retained therein once a disconnect occurred.

After a connection is restored, in step 210, the modem 86 transmits a reconnect message signal to the SUSPENDED state processing component 28B and provides reconnect control signals to the SUSPENDED state processing component 88B indicating restoration of the communication link. Upon being shifted to control operations, the SUSPENDED state processing component 28B waits to receive such signals. Upon reception of the reconnect signals, in step 212, the WAKE-UP state processing components 88C and 28C begin performing operations and the SUSPENDED state processing components 28B and 88B cease to perform operations.

During the performance of WAKE-UP state operations, data packet transfer between the transmitter 20 and receiver 80 is synchronized to provide for transmission of only those data packets which are not currently stored in the buffer 84 of the receiver 80. In particular, the WAKE-UP state processing component 88C retrieves the value in PNC from the RAM memory 82, and, based on the value of PNC, provides that a status message signal be transmitted to the transmitter 20 for routing to the WAKE-UP state processing component 28C. The status message signal contains data identifying those data packets previously received and stored at the

buffer 84 before the disconnect occurred. For example, if data packet DP 30 has been transmitted just before a disconnect occurs and the value of PNC is and remains equal to 20, the previously transmitted data packets DP 21, DP 22, . . . DP 29 would not have been correctly received and stored in the buffer 84. In this case, the status message signal would include information indicating that data packets DP 21, DP 22, . . . DP 29 have not been received and stored in the buffer 84. After the processor 28 receives and processes a status message signal, in step 216, the processor 28 begins performing NORMAL state operations.

In step 216, the WAKE-UP state processing component 28C, based on the value of PNC represented in the status message signal, directs the modem 22 to begin transmitting those data packets of the data file not yet received and stored at the buffer 84 starting from that data packet having a sequence number corresponding to the value of PNC plus one. The data packet transmissions may involve retransmission of previously transmitted data packets not correctly received at the receiver 80 as well as the original transmission of data packets. In step 218, the NORMAL state processing component 88A begin to perform operations upon detecting the receipt of a retransmitted data packet from the transmitter 20.

It is to be understood that, although the receiver 80 may preferably establish the data link layer communication link by transmitting status message signals to the transmitter 20 after a disconnect occurs, it may be more convenient in some situations for the transmitter 20 to start the communication. For example, in the case of an asymmetric protocol like AIRMAIL, described in detail in E. Ayanoğlu, S. Paul, T. F. LaPorta, K. K. Sabnani, R. D. Gitlin, AIRMAIL: A Link-layer Protocol for Wireless Networks, Wireless Networks Vol. 1, No. 1, pp. 47-60, 1995; U.S. application Ser. No. 08/282,254, filed Jul. 29, 1994, both of which are incorporated by reference herein, a transmitting base station communication device in a mobile network should start the data link-layer communication because only the transmitter has the capability to detect re-establishment of the link and the receiving mobile host in AIRMAIL cannot detect the re-establishment of a link. For example, in AIRMAIL, if the base station continues to transmit during the disconnect, a poll message will be transmitted to a mobile receiving communication device requesting its status.

Alternatively, if a mobile receiving device changes from one base station to another while the mobile device is performing SUSPENDED state operations, the mobile device, on sensing a different base station, must initiate the communication with the new base station by sending a handoff request and following the steps associated with a regular handoff, as described in the AIRMAIL protocol cited above.

In another aspect of the present invention, an additional protocol, or recovery protocol layer, may be suitably included in the protocol layer hierarchy of a communication operating system for controlling data transfer applications for avoiding retransmission of data packets correctly received and stored in a buffer when a disconnect occurs. Similar to the recovery module described above, the recovery protocol layer suitably comprises programmed instructions which provide that data packets correctly received at a receiver are retained in the receiver memory in the event of a disconnect while attempts are made to restore the communication link. A record of the sequence number of correctly received and stored data packets is maintained, based on data packets stored in the buffer, for synchronizing transmission of data packets after a connection is restored.

The recovery protocol layer is suitably included in the hierarchy of an operating system to provide for control of processing at a protocol layer, such as an end-to-end protocol layer, which may become destabilized in the event of a disconnect. For example, in the exemplary communication operating system described above for the system 10, the recovery protocol layer may be located in the hierarchy below the application layer and above the transport layer. The transport protocol layer suitably performs end-to-end operations and the application protocol performs application specific operations, such as transferring of files and sending e-mail messages.

The addition of a new and separate protocol layer to a communication operating system, rather than the modification of an existing protocol layer, provides the advantage of ease in implementation, as existing protocol layers would not require modification. However, an increased cost is incurred when disconnects are processed at an additional higher level protocol layer. An additional protocol layer increases overhead costs by requiring an additional header in a message signal for implementing the respective protocol operations. The utilization of an additional header increases the overhead associated with data transmission by consuming more bandwidth, causing decreased throughput.

FIG. 4 is a state diagram which illustrates operations that may be suitably performed by the transmitter 20 and receiver 80, respectively, for controlling data packet transfer activities in the system 10 during a communication session between the former and latter in the event of a disconnect in accordance with the present inventive technique of including a recovery protocol layer for stabilizing another protocol layer.

The recovery protocol layer provides that the transmitter 20 operate in a TRANSMIT state and the receiver 80 operates in a RECEIVE state when data packet transfer is occurring normally, in other words, when a disconnect has not occurred. The transmitter 20 operates in SUSPENDED state and the receiver 80 operates in SUSPENDED states when a disconnect occurs. Upon restoration of the communication link, the receiver 80 operates in the RECEIVE state 400, while the transmitter 20 initially operates in a RETRANSMIT state for retransmitting originally transmitted data packets which were not correctly received and saved by the receiver 80. After performing these retransmissions, the transmitter 20 transitions from the RETRANSMIT state to the TRANSMIT state, where original data packet transmission continues from the data packet having a sequence number which is one greater than the sequence number of the last data packet transmitted before the disconnect occurred.

The recovery protocol layer is suitably implemented using the checkpointing window transmission technique for monitoring and controlling data transfer operations in the event of a disconnect for avoiding retransmission of data packets which already have been correctly received and stored in the buffer of a receiving device. Checkpointing, described in detail in S. Aggarwal, K. Sabnani, B. Gopinath, A New File Transfer Protocol, *AT&T Technical Journal*, Vol. 64, No. 10, pp. 2387-2411, December 1985, incorporated by reference herein, provides for the transmission of data packets during a communication session using a sliding window of transmission. A window of transmission is defined by a lower end value, L, and a higher end value H, where the values of L and H are used for referring to sequence numbers of data packets in the data file being transferred. The size of a transmitting window, W, defines the maximum difference between L and H and is computed based on a predefined checkpointing

interval, which is a measure of how much temporary data loss a specific data transfer application can tolerate during the state of disconnection. During a communication session, data packets having sequence numbers between L and H are continuously transmitted and periodic transmission of acknowledgement message signals from a receiver to the transmitter must occur for indicating continuous reception of data packets. For example, the transmitter 20 continues to transmit data packets until data packets having sequence numbers between H and L, inclusive, have been transmitted. An acknowledgement message signal is suitably transmitted periodically after, for example, 8 data packets have been received in succession, to ensure continuous transmission of data packets. As the transmitter 20 receives acknowledgement signals from the receiver 80, L is continuously updated for sliding the transmitting window forward. The acknowledgement signals include data representative of the last data packet received at the receiver, and the transmitter uses this data for updating the value of L, such that transmission of data packets according to the size of the window W continues. If a disconnect occurs during a communication, data packets are no longer correctly received and stored at the receiver to provide for the transmission of acknowledgement signals in ordinary course, and in turn results in the cessation of original data packet transmission.

In accordance with the present invention, in the normal course of data transfer operations where a disconnect has not occurred, the processor 28 of the transmitter 20 would operate in the TRANSMIT state and the processor 88 of the receiver 80 would operate in the RECEIVE state 400, where the receiver 80 transmits acknowledgement message signals to the transmitter 20 indicating the sequence number of the last data packet correctly received and stored in the buffer 84. A disconnect, however, causes these acknowledgement signals not to be transmitted to and received in ordinary course at the transmitter 20. As more fully discussed below, when a disconnect occurs, the recovery protocol layer controls operations at the transmitter 20, which shift first to the SUSPENDED state, and then from the SUSPENDED state to RETRANSMIT state when a reconnect occurs before control ultimately shifts back to the TRANSMIT state. In addition, as also more fully discussed below, when a disconnect occurs, the recovery protocol layer controls operations at the receiver 80, which shift to the SUSPENDED state 420.

FIG. 5A shows an exemplary embodiment of the transmitter 20 which may be utilized for implementing a recovery protocol layer as an independent protocol layer in accordance with the present inventive technique. Referring to FIG. 5A, the transmitter 20 may suitably comprise the interconnection of the RAM 22, the modem 26 and the buffer 24, as described above with reference to FIG. 2. The processor 28 comprises the functional processing blocks of a TRANSMIT state processing component 28D, a SUSPENDED state processing component 28E and a RETRANSMIT state processing component 28F for performing the operations described below with reference to FIGS. 6A, 6B, 6C and 6D. Current values of the lower end, L, and the higher end, H, of the transmission window W are stored in the RAM 82. In addition, data for indicating the sequence numbers of data packets which are to be retransmitted from the transmitter 20 during the RETRANSMIT state, as discussed below, are stored at memory location Ltmp in the RAM 22.

FIG. 5B shows an exemplary embodiment of the receiver 80 which may be utilized for implementing a recovery protocol layer as an independent protocol layer in accordance with the present inventive technique. Referring to FIG. 5B, the receiver 80 may suitably comprise the interconnection of the RAM 82, the modem 86 and the buffer 84, as described above with reference to FIG. 2. The processor 88 comprises the functional processing blocks of a RECEIVE state processing component 88D, a SUSPENDED state processing component 88E and a RETRANSMIT state processing component 88F for performing the operations described below with reference to FIGS. 6A, 6B, 6C and 6D. Current values of the lower end, L, and the higher end, H, of the transmission window W are stored in the RAM 22. In addition, data for indicating the sequence numbers of data packets which are to be retransmitted from the receiver 80 during the RETRANSMIT state, as discussed below, are stored at memory location Rtmp in the RAM 82.

dance with the present inventive technique. Referring to FIG. 5B, the receiver 80 may suitably comprise the interconnection of the RAM 82, the modem 86 and the buffer 84, as described above with reference to FIG. 2. The processor 88 suitably comprises the functional processing blocks of a RECEIVE state processing component 88D and a SUSPENDED state processing component 88E for performing the operations described below with reference to FIGS. 4, 7A, 7B and 7C. Data representative of the sequence numbers of data packets being received and stored in the buffer 84 and data representations of the data packet sequence number for which an acknowledgement message signal was last transmitted are stored in memory locations PNO and Lpkt, respectively, in the RAM 82. In addition, data representative of the successive reception of data packets are stored in memory location COUNT in the RAM 82 for purposes of comparison with the size of the window W and providing for original data packet transmission.

Reference to data values which are used in processing and are stored in the RAM of a communication device for performing the present inventive technique are identified below using identical designations utilized in the method 200 above when reference is being made to similar data information. For purposes of illustration, it is assumed that original sequential data packet transmission has commenced such that the transmitter 20 is operating in the TRANSMIT state and the receiver 80 is operating in the RECEIVE state 400. Further, it is assumed that the size of the checkpointing window W is equal to 16.

The operations performed at the transmitter 20 and receiver 80 are mutually dependent, as the occurrence of a disconnect affects operation at the former and latter. While a disconnect is not present, the TRANSMIT state processing component 28D performs operations in the TRANSMIT state, which includes the simultaneous performance of a data packet transmission routine 307 and a reception of acknowledgement routine 314, which are described in further detail in FIGS. 6A and 6B, respectively. The data packet transmission routine 307 and the acknowledgement routine 314 are simultaneously performed by the TRANSMIT state processing component 28D so long as a disconnect has not occurred.

Referring to FIG. 6A, for the transmit routine 307, in step 308, the TRANSMIT state processing component 28D first computes the difference between the values of H and L which are stored in the RAM 22 and compares the result to the value of W. If $H-L=W$, then a disconnect has been invoked. As discussed above, a disconnect may be invoked by the modem 86 of the receiver 80 or by routers in the network, as is conventional in communication system operation. As discussed below, when a disconnect occurs, further operation in the TRANSMIT state ceases, and operation by the SUSPENDED state processing component 28F in the SUSPENDED state begins. If $H-L$ is not equal to W, in step 309, the TRANSMIT state processing component 28D determines whether an acknowledgment message signal, ACK(Lr), has been received from the receiver 80. Lr represents the value of the sequence number of the last data packet of a group of transmitted data packets which was correctly received and saved in the buffer memory 84 of the receiver 80. If an acknowledgement message signal was received, in step 310, the TRANSMIT state processing component 28D sets the value of L equal to the value of Lr. Step 310, thus, enables the transmitter 20 to maintain a record of the data packets which have been correctly received and stored at the receiver 80. For each acknowledgement message signal received at the transmitter 20, the

window of transmission slides forward to provide for continuous data packet transmission, as described more fully below with reference to the transmission routine 314. The routine 307 is continually re-executed as long as a disconnect condition is not detected at step 308.

Referring to FIG. 6B, for acknowledgement routine 314, in step 315, the TRANSMIT state processing component 28D compares the difference between H and L as in step 308 above for detecting a disconnect. If a disconnect is not detected, in step 316, the TRANSMIT state processing component 28D compares the difference between the values of H and L to the checkpoint value W.

If $H-L < W$, then in steps 317A, 317B, 317C and 317D, the TRANSMIT state processing component 28D provides for the transmission of additional data packets to the receiver 80 until the number of data packets which have been currently transmitted equals the value of the checkpointing transmission window W. In step 317A, the TRANSMIT state processing component 28D sets the value of a location LOOP in the memory 22 equal to zero. Then, in step 317B, the TRANSMIT state processing component 28D determines whether the sequence number corresponding to data packet DP ($H+1+LOOP$) is greater than the sequence number corresponding to data packet DP ($L+W$). If this is not true, in step 317C, the TRANSMIT state processing component 28D provides for the transmission of data packet DP ($H+1+LOOP$). Then, in step 317D, the TRANSMIT state processing component 28D increments the value of LOOP in the memory 22 by one. Step 317A is performed after step 317D. However, if the inequality condition at step 317 is true, then step 319 is executed. In other words, steps 317A, 317B, 317C and 317D provide that data packets DP ($H+1$) to DP ($L+W$) are transmitted from the transmitter 20. For example, if at step 317, H is equal to 23, L is equal to 15 and W is equal to 16, the TRANSMIT state processing component 28D would provide for the transmission of data packets DP 16 to DP 31. Finally, in step 319, the TRANSMIT state processing component 28D updates the value of H to the new higher end value of the checkpointing window based on the most recent transmission of data packets in step 317C, such that H is set equal to the sum of L and W.

Referring to FIG. 4, while a disconnect has not been detected, the RECEIVE state processing component 88D of the receiver 80 performs a data packet receive counting routine 407 and an acknowledgement signal transmission routine 414, which are more fully discussed below with reference to FIGS. 7A and 7B, respectively. The routines 407 and 414 are performed as appropriate so long as a disconnect does not occur.

Referring to FIG. 7A, for the counting routine 407, when a disconnect is not present, in step 408, the RECEIVE state processing component 88D determines whether the value of COUNT is less than W/2. If yes, in step 410, the receiver 80 continues to receive data packets from the transmitter 20 and, as is conventional in the art, the RECEIVE state processing component 88D stores correctly received data packets in the buffer 84. The RECEIVE state processing component 88D, further, tracks the succession of data packets being correctly received by updating the value of PNO in the RAM 82 with the sequence number of the last correctly received and stored data packet. Then in step 412, the RECEIVE state processing component 88D increments the value of COUNT by one for monitoring the successive reception and storage of data packets in the buffer 84.

Referring to FIG. 7B, for the acknowledgement routine 414, when a disconnect is not present, in step 415, the

RECEIVE state processing component 88D determines if the value of COUNT is equal to W/2. If yes, in step 417, the RECEIVE state processing component 88D performs operations similar to those of step 410 of the routine 407 for receiving, storing and tracking data packets transmitted by the transmitter 20 and received at the receiver 80. Then in step 418, the RECEIVE state processing component 88D transmits an acknowledgement message signal, Ack(PNO), to the transmitter 20, where PNO indicates the sequence number of the last data packet correctly received and stored in the buffer 84. Then in step 419, the RECEIVE state processing component 88D sets the value of COUNT equal to zero and sets the value of Lpkt equal to PNO. For each acknowledgement message signal transmission, therefore, an indication is provided to the transmitter 20 that W/2 data packets have been received and stored in the buffer 84.

As described above, transmission of acknowledgement message signals in ordinary course indicates a disconnect has not occurred. The receiver 80, thus, remains in the RECEIVE state 400 and the transmitter 20 remains in the TRANSMIT state as long as a disconnect is not invoked. In contrast, when a disconnect occurs, the transmitter 20 would not continue to receive acknowledgement message signals from the receiver 80 as COUNT would not continue to be incremented in step 412 of the routine 407. As the transmitter 20 does not receive acknowledgement message signals from the receiver 80, steps 309 and 310 will not be performed and L will not be updated. However, the transmitter 20 would continue to transmit data packets per routine 314 until H=L+W. When the condition H=L+W is true, however as in steps 308 and 315, the transmitter 20 begins operating in the SUSPENDED state. In the SUSPENDED state, the transmitter 20 does not transmit data packets to the receiver 80. In addition, during the SUSPENDED state, the modems 26 and 86 of the transmitter 20 and the receiver 80, respectively, suitably attempt to restore a communication link between the former and latter in accordance with standard techniques. While in the SUSPENDED state, the SUSPENDED state processing component 28E performs a check-for-reconnect routine 325, as shown in FIG. 6C. Referring to FIG. 4, while in the SUSPENDED state 420, the SUSPENDED state processing component 88E performs a check-for-reconnect routine 425, as shown in FIG. 7C.

The SUSPENDED state processing component 88E operates in the SUSPENDED state 420 when a disconnect has occurred, and provides that all data packets previously correctly received and stored in the buffer 84 as part of the requested data file transfer continue to be stored in the buffer 84, despite the occurrence of the disconnect.

Referring to FIG. 7C, for the routine 425, in step 427, the SUSPENDED state processing component 88D detects whether the communication link to the transmitter 20 has been re-established from the receiver 80. For example, the modem may be able to establish a connection after several attempts are made, and then transmit message signals to the SUSPENDED state processing component 88E of the receiver 80 indicating that a reconnect has occurred. If such message signals are not detected, the SUSPENDED state processing component 88E continues to operate in the SUSPENDED state 420. Otherwise, in step 429, the SUSPENDED state processing component 88E transmits an acknowledgement message signal, ACK(Lpkt), to the transmitter 20, where Lpkt represents the sequence number of the last data packet in sequence received and stored at the buffer 84 at the time the last acknowledgement message signal was transmitted at step 418. For example, the last packet H transmitted by the transmitter 20 may have a sequence

number that exceeds Lpkt, which indicates to the transmitter 20 that there are H-Lpkt outstanding data packets that the receiver 80 did not receive as a result of the disconnect. In a preferred embodiment, these few, if any, data packets are retransmitted, as more fully discussed below in connection with FIG. 6D. When H is equal to Ltemp, the transmitter 20 shifts operation from the RETRANSMIT state to the TRANSMIT state. After performing step 429, the receiver 80 then operates in the RECEIVE state 400.

Referring to FIG. 6C, for the routine 325, in step 327, the SUSPENDED state processing component 28E detects whether an acknowledgement message signal, Ack(Lr), has been received from the receiver 80. The reception of an acknowledgement message signal at the transmitter 20 which has been transmitted from the receiver 80 indicates that a connection link between the transmitter 20 and receiver 80 has been restored. If an acknowledgement message signal has been received, in step 329, the SUSPENDED state processing component 28E sets the value of L equal to the value Lr contained in the acknowledgement message signal at step 327. Further, the SUSPENDED state processing component 28E sets Ltemp equal to the value of L for facilitating retransmission of data packets previously transmitted but not received and stored at the receiver 80. If an acknowledgement message signal has not been received at step 327, the SUSPENDED state processing component 22 continues to perform the SUSPENDED state operations, as a connection link has not been restored.

The RETRANSMIT state processing component 28F performs operating in the RETRANSMIT state once a reconnect is detected and, in particular, performs a retransmit data packet routine 335, shown in FIG. 6D. The Ack(Lr) message signal which was received at step 327 of the routine 325 indicates that a connection link to the transmitter 20 has been restored. Referring now to FIG. 6D, in step 337, the RETRANSMIT state processing component 28F determines whether Ltemp is less than H. This step ensures the retransmission of all data packets which have been previously transmitted before the disconnect occurred, but not necessarily received and stored in the buffer 84. If yes, in step 339, the RETRANSMIT state processing component 28F provides control signals to the modem 26 to provide for the retransmission of the data packet whose sequence number corresponds to DP (Ltemp+1), as the data packet DP (Ltemp) was the last correctly received and stored data packet. In step 341, the RETRANSMIT state processing component 28F sets Ltemp equal to Ltemp+1. After step 341, the RETRANSMIT state processing component 28F performs step 337. In other words, the RETRANSMIT state processing component 28F continues to perform steps 337 to 341 until all data packets with sequence numbers less than or equal to H have been retransmitted. These data packets are retransmitted to the receiver 80 because they may not have been correctly received and stored at the receiver 80 based on an original transmission that occurred before the disconnect.

For example, if L=7, H=23 and W=16 and an acknowledgement message signal Ack(19) is received by the transmitter 20 in step 327, the transmitter 20 would retransmit data packets DP 11 through DP 23 while in the RETRANSMIT state. After these previously transmitted data packets are retransmitted, Ltemp would have been incremented to H. When this occurs, the transmitter 20 begins operating in the TRANSMIT state.

Thus, the continued storage of data packets in the buffer 84 during a disconnect enables the transmitter 20 to synchronize the retransmission of data packets and the original

15

transmission of data packets of a data file for avoiding unnecessary retransmissions of data packets already stored in the buffer 24. As a result, improved throughput efficiency is obtained. An entire data file of data packets may be stored in a buffer more quickly to allow for quicker data transfer application processing. For example, for a data file transfer of a map of an area, data for reconstructing the map would be completely received at an earlier point in time, thereby allowing the map to be usefully displayed to the user providing added convenience. It is to be understood that the embodiments and variations shown and described above are illustrative of the principles of this invention only and that various modifications may be implemented by those skilled in the art without departing from the scope and spirit of the invention.

We claim:

1. A method for controlling data transfer operations comprising the steps of:

- (a) initiating a data file transmission from a first communication device to a second communication device over a communication link, said data file comprising a plurality of sequentially numbered data packets, said transmission involving the sequential transmission of said data packets;
 - (b) storing data packets correctly received at a memory in said second communication device;
 - (c) maintaining a record in the second communication device memory of the sequence numbers of those data packets stored in the second communication device memory; and
 - (d) in response to receiving a signal indicative of a disconnect in the communication link, retaining in the second communication device memory those data packets stored in the second device memory before the disconnect occurred, such that the first communication device need transmit to the second communication device, once the communication link is re-established, only those data packets of the data file not previously stored in the second communication device memory.
2. The method of claim 1, further comprising the step of:
- (e) transmitting a status message signal from the second communication device to the first communication device, once the communication link is re-established, said status message signal indicating the sequence number of the last data packet in sequence received and stored in the second communication device memory.
3. The method of claim 2, further comprising the step of:
- (f) retransmitting to the second communication device from the first communication device the data packets previously transmitted from the first communication device but not stored in the second communication device memory.
4. The method of claim 2, further comprising the step of:
- (f) transmitting to the second communication device from the first communication device data packets not previously transmitted from the first communication device to the second communication device.

5. The method of claim 1, wherein programmed instructions corresponding to the operations performed in steps (a) through (d) are included in an existing protocol layer of a communications system, said existing protocol layer performing communication operations as well as the operations performed in steps (a) through (d), said system comprising a plurality of protocol layers.

6. The method of claim 1, wherein programmed instructions corresponding to the operations performed in steps (a)

16

through (d) are included in an independent protocol layer of a communications system, said independent protocol layer operative to perform only the operations of steps (a) through (d), said system comprising a plurality of protocol layers.

7. A method for controlling data transfer operations comprising the steps of:

initiating a data file transmission from a first communication device to a second communication device over a communication link, said data file comprising a plurality of sequentially numbered data packets, said transmission involving the sequential transmission of said data packets;

maintaining checkpointing window transmission values in a memory of said first communication device for controlling the transmission of said data packets, said window transmission values including a lower and higher bound value and a window size value;

transmitting data packets in sequence as windows of information according to the lower and upper bound transmission values and the window size value;

monitoring the successive reception of the data packets at the second communication device;

storing data packets correctly received at said second communication device in a memory of said second communication device;

storing data concerning the successive reception of the data packets received in a counter location in the second communication device memory;

transmitting acknowledgement signals periodically from the second communication device to the first communication device for indicating the reception and storage of said data packets, said acknowledgement signals being transmitted when the successive count value is equal to a fraction of the window size value;

updating the value of the lower bound value for each acknowledgement signal received at the first communication device;

ceasing data packet transmission when a disconnect occurs;

receiving at the first communication device an acknowledgement signal from the second device indicating re-establishment of the link; and

retransmitting data packets in sequence, after the first communication device received the link re-establishment acknowledgement signal, starting from the data packet having a sequence number one greater than the last data packet stored in the second device memory corresponding to the last data packet transmitted before the disconnection occurred.

8. The method of claim 7, further comprising the steps of: receiving a signal at the second device indicating the re-establishment of the link; and,

transmitting a sequence number acknowledgement signal from the second device to the first communication device indicating the sequence number of the last packet in sequence received and stored in the second device memory.

9. The method of claim 7, wherein the operations of said method are performed as part of an independent protocol layer included in a communication operating system, said system comprising a plurality of protocol layers.

10. A system for controlling data transfer operations comprising:

a first communication device comprising at least one processor and a memory;

17

a second communication device comprising at least one processor and a memory, said first communication device being linked to said second device over a communication link when said first communication device performs the operation of initiating a data file transmission to the second communication device, said data file comprising a plurality of sequentially numbered data packets, said transmission involving the sequential transmission of said data packets;

receiver for receiving at said second communication device said data packets transmitted from said first communication device;

a memory for storing said received data packets in said second communication device memory;

said second communication device memory maintaining a record of the sequence numbers of those data packets stored in the second communication device memory, wherein said processor of said second communication device in response to receiving a signal indicative of a disconnect in the communication link, retains in the second communication device memory those data packets stored in the second communication device memory before the disconnect occurred, such that the first communication device need transmit to the second communication device, once the communication link is re-established, only those data packets of the data file not previously stored in the second communication device memory.

11. The system of claim 10, wherein said processor of said second communication device further performs the operation of:

generating transmitting control signals to provide for transmission of a status message signal to the first communication device, once the communication link is re-established, indicating the last data packet in sequence stored in the second communication device memory.

12. The system of claim 11, wherein said processor of said first communication device further performs the operation of:

generating retransmitting control signals to provide for retransmission of the data packets previously transmitted from the first communication device but not stored in the second communication device memory.

13. The system of claim 10, wherein said processor of said first communication device further performs the operation of:

generating transmitting control signals to provide for the transmission of data packets not previously transmitted from said communication first device.

14. The system of claim 10, wherein programmed instructions corresponding to the operations performed by the processors of said first and second communication devices, respectively, are included as part of an existing protocol layer of a communication system, said existing protocol layer performing communication operations as well as the steps of initiating a data file transmission from the first communication device to the second communication device, receiving data packets at the second communication device, storing received data packets in the memory of the second communication device, maintaining in the memory of the second communication device a record of the sequence numbers of the data packets stored in the second communication device memory, and in response to receiving at the second communication device a signal indicative of a disconnect in the communication link, retaining in the second

18

communication device memory the data packets stored in the second communication device memory before the disconnect occurred, said instructions being stored in the memories of said first and second communication devices, respectively.

15. The system of claim 10, wherein programmed instructions corresponding to the operations performed by the processors of said first and second communication devices, respectively, are included as part of an independent protocol of a communication system, said independent protocol layer performing the steps of initiating a data file transmission from the first communication device to the second communication device, receiving data packets at the second communication device, storing received data packets in the memory of the second communication device, maintaining in the memory of the second communication device a record of the sequence numbers of the data packets stored in the second communication device memory, and in response to receiving at the second communication device a signal indicative of a disconnect in the communication link, retaining in the second communication device memory the data packets stored in the second communication device memory before the disconnect occurred, said instructions being stored in the memories of said first and second communication devices, respectively.

16. A system for controlling data transfer operations, comprising:

a first communication device comprising at least one processor and a memory;

a second communication device comprising at least one processor and a memory, said first communication device being linked to said second communication device over a communication link when said first communication device performs the operation of initiating a data file transmission to the second communication device, said data file comprising a plurality of sequentially numbered data packets, said transmission involving the sequential transmission of said data packets;

said first communication device processor maintaining checkpointing window transmission values in said first communication device memory for controlling the transmission of said data packets, said window transmission values including a lower and higher bound value and a window size value;

said first communication device transmitting data packets in sequence according to the lower and upper bound values;

said second communication device monitoring the successive reception of data packets;

said second communication device processor storing data concerning the successive reception of the data packets in a counter location in the second communication device memory;

said second communication device periodically transmitting acknowledgement signals to the first communication device indicating the reception and storage of said data packets, said acknowledgement signals being transmitted when the successive count value is equal to a fraction of the window size value;

said first communication device processor updating the value of the lower bound value for each acknowledgement signal received;

said first communication device ceasing data packet transmission when a disconnect occurs;

said first communication device receiving an acknowledgement signal from the second communication device indicating re-establishment of the link; and

19

said first communication device retransmitting data packets in sequence, after the first communication device received the link re-establishment acknowledgement signal starting from the data packet having a sequence number one greater than the last data packet stored in the second communication device memory corresponding to the last data packet transmitted before the disconnection occurred.

17. The system of claim 15, wherein said second communication device receives a signal indicating the re-establishment of the link and transmits a sequence num-

20

ber acknowledgment signal to the first communication device indicating the sequence number of the last packet in sequence received and stored in the second communication device memory.

18. The system of claim 15, wherein the operations of said system are performed as part of an independent protocol layer included in a communication operating system, said system comprising a plurality of protocol layers.

* * * * *



US006463474B1

(12) United States Patent
Fuh et al.**(10) Patent No.: US 6,463,474 B1**
(45) Date of Patent: Oct. 8, 2002**(54) LOCAL AUTHENTICATION OF A CLIENT AT A NETWORK DEVICE****(75) Inventors:** **Tzong-Fen Fuh**, Fremont; **Serene H. Fan**, Palo Alto; **Diheng Qu**, Santa Clara, all of CA (US)**(73) Assignee:** **Cisco Technology, Inc.**, San Jose, CA (US)**(*) Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.**(21) Appl. No.: 09/347,433****(22) Filed: Jul. 2, 1999****(51) Int. Cl.⁷ G06F 15/173****(52) U.S. Cl. 709/225; 709/229; 709/232; 713/201****(58) Field of Search 709/229, 225, 709/223, 232; 713/200, 201****(56) References Cited****U.S. PATENT DOCUMENTS**

5,991,807 A * 11/1999 Schmidt et al. 709/225
 6,182,142 B1 * 1/2001 Win et al. 709/229
 6,219,706 B1 * 4/2001 Fan et al. 709/225
 6,233,576 B1 * 5/2001 Lewis 707/9
 6,233,618 B1 * 5/2001 Shannon 709/229
 6,292,798 B1 * 9/2001 Dockter et al. 707/9

6,292,904 B1 * 9/2001 Broomhall et al. 714/1

* cited by examiner

Primary Examiner—Glenton B. Burgess
Assistant Examiner—Kimberly D Flynn
 (74) *Attorney, Agent, or Firm*—Hickman Palermo Truong & Becker LLP

(57) ABSTRACT

A method and apparatus that provide network access control are disclosed. In one embodiment, a network device is configured to intercept network traffic initiated from a client and directed toward a network resource, and to locally authenticate the client. Authentication is carried out by comparing information identifying the client to authentication information stored in the network device. In one embodiment, an authentication cache in the network device stores the authentication information. If the client identifying information is authenticated successfully against the stored authentication information, the network device is dynamically re-configured to allow network traffic initiated by the client to reach the network resource. If local authentication fails, new stored authentication is created for the client, and the network device attempts to authenticate the client using a remote authentication server. If remote authentication is successful, the local authentication information is updated so that subsequent requests can authenticate locally. As a result, a client may be authenticated locally at a router or similar device, reducing network traffic to the authentication server.

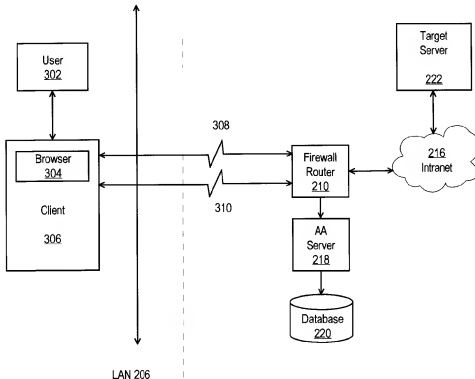
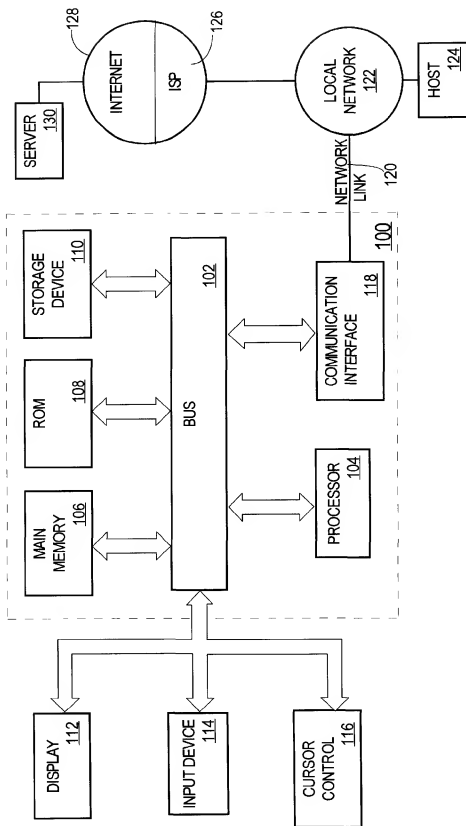
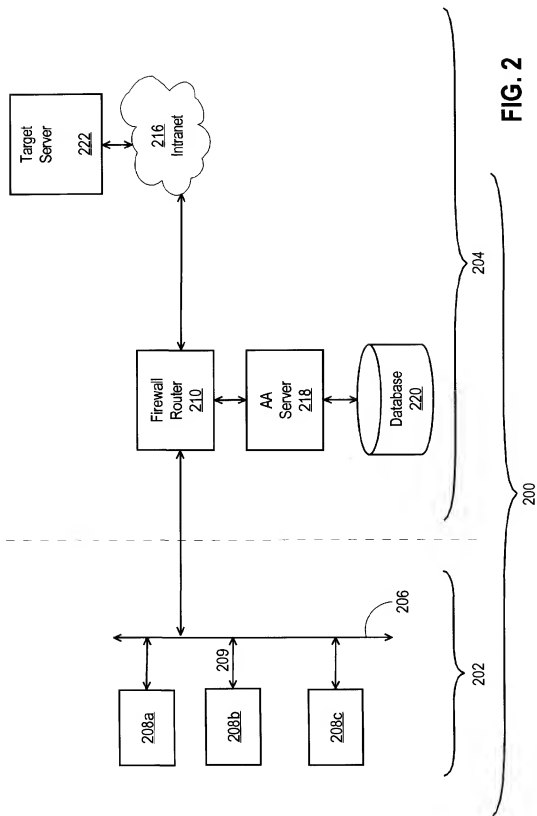
21 Claims, 9 Drawing Sheets

FIG. 1



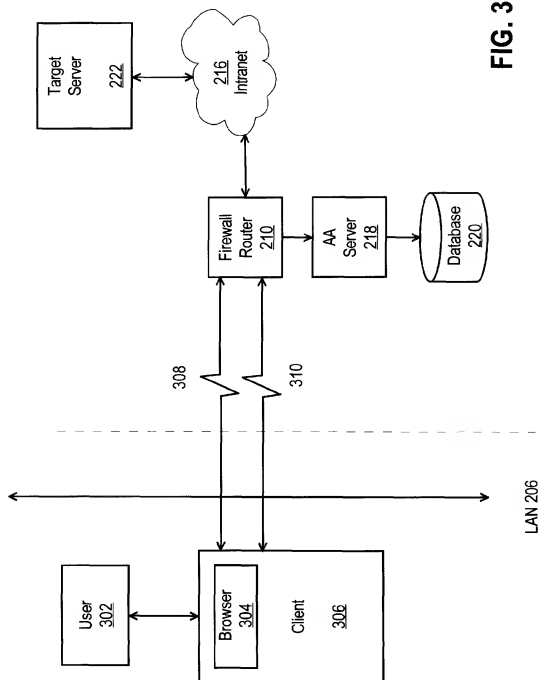
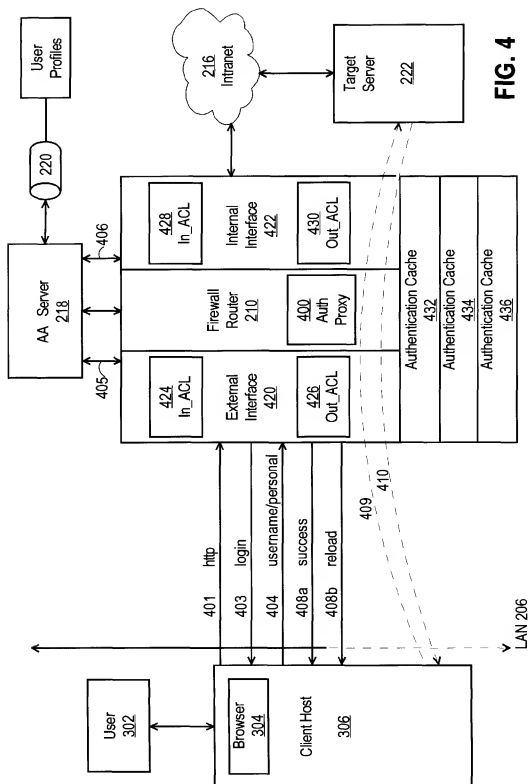


FIG. 3



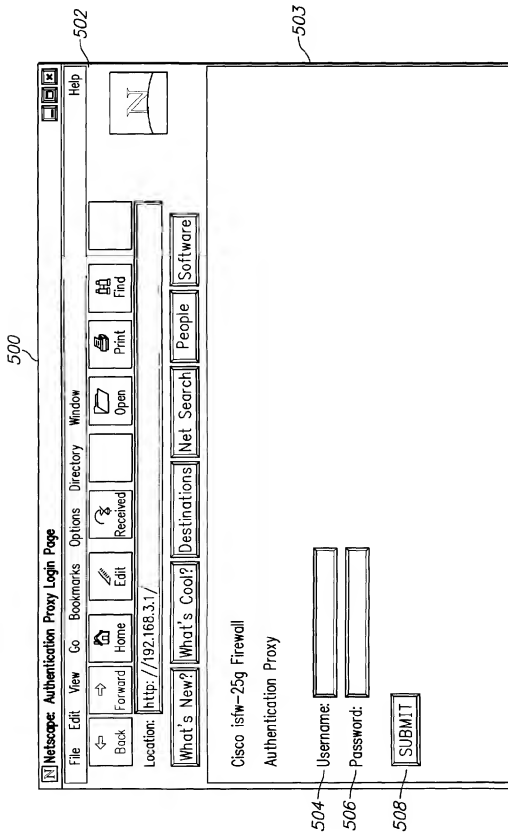


FIG. 5A

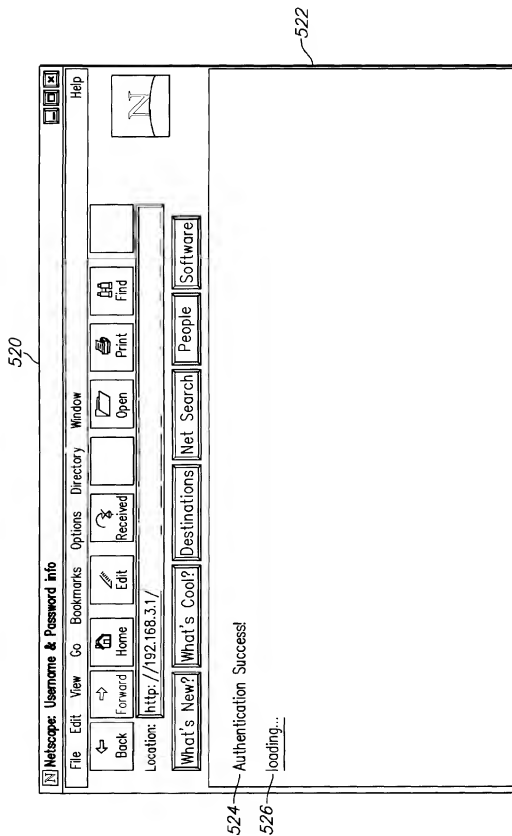


FIG. 5B

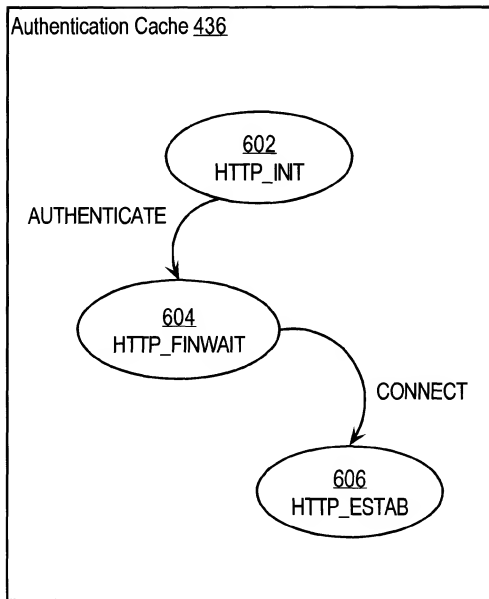
FIG. 6

FIG. 7A

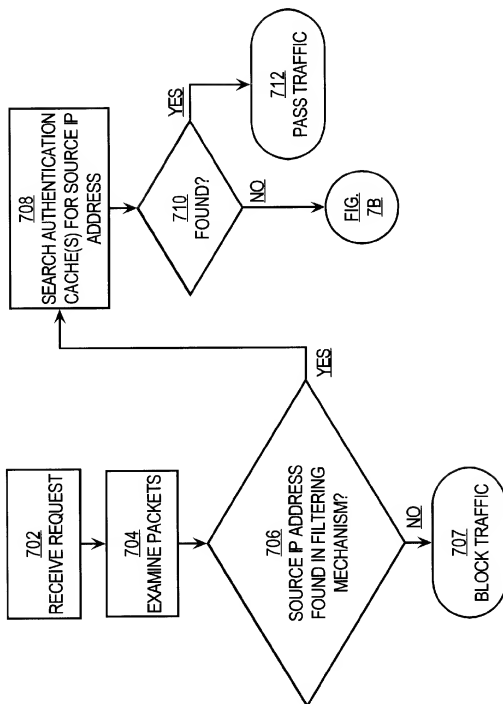
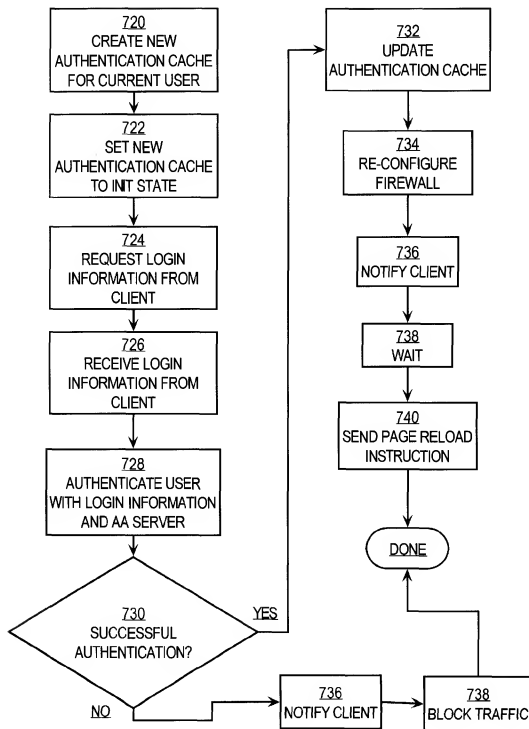


FIG. 7B



1

LOCAL AUTHENTICATION OF A CLIENT AT A NETWORK DEVICE

FIELD OF THE INVENTION

The present invention generally relates to management of computer networks, and relates more specifically to authentication and authorization mechanisms for network devices such as routers and firewalls.

BACKGROUND OF THE INVENTION

Computer users often access information, computer files, or other resources of computer networks from locations that are geographically or logically separate from the networks. This is referred to as remote access. For example, a user of a host or client that is part of a local area network ("LAN") may want to retrieve information that resides on a computer that is part of a remote network. Before a user can gain access to that computer, the user must first obtain permission to do so. In the interest of data integrity, and data confidentiality, many computer networks have implemented integrity and access control mechanisms to guard against unwanted network traffic or access by unauthorized users. On the other hand, a corporation may institute policies that restrict its employees from accessing certain web sites on the internet while using the corporation's computer resources. For example, Corporation C may disallow access to pornographic web sites. Corporation C's access control mechanism would prevent the employees from accessing such sites.

An example of an access control mechanism is a server that implements authentication, authorization, and accounting ("AAA") functions. Authentication is the process of verifying that the user who is attempting to gain access is authorized to access the network and is who he says he is. Generally, after authentication of a user, an authorization phase is carried out. Authorization is the process of defining what resources of the network an authenticated user can access.

Several authentication and authorization mechanisms are suitable for use with operating systems that are used by network devices, such as the Internetworking Operating System ("IOS") commercially available from Cisco Systems, Inc. However, most prior authentication and authorization mechanisms are associated with dial-up interfaces, which can create network security problems. In a dial-up configuration, a remote client uses a telephone line and modem to dial up a compatible modem that is coupled to a server of the network that the remote client wishes to access. In another dial-up configuration, a remote client first establishes a dial-up connection to a server associated with an Internet Service Provider, and that server then connects to the network server through the global, public, packet-switched internetwork known as the Internet. In this configuration, the network server is coupled directly or indirectly to the Internet.

Unfortunately, information requests and other traffic directed at a network server from the Internet is normally considered risky, untrusted traffic. An organization that owns or operates a network server can protect itself from unauthorized users or from unwanted traffic from the Internet by using a firewall. A firewall may comprise a router that executes a "packet filter" computer program. The packet filter can selectively prevent information packets from passing through the router, on a path from one network to another. The packet filter can be configured to specify which

2

packets are permitted to pass through the router and which should be blocked. By placing a firewall on each external network connection, an organization can prevent unauthorized users from interfering with the organization's network of computers. Similarly, the firewall can be configured to prevent the users of the organization's network of computers from accessing certain undesirable web sites on the Internet.

One common method of remote access using the Internet is telnet, a protocol used to support remote login sessions that defines how local and remote computers talk to each other to support a remote login session. "Telnet" is also the name of a remote login program commonly used in networks based on Transmission Control Protocol/Internet Protocol ("TCP/IP"), a set of protocols that define how communications occur over the Internet. Past authentication and authorization mechanisms were produced to work with firewalls in the context of telnet. An example of an authentication and authorization mechanism that works with telnet is "Lock and Key" for IOS, commercially available from Cisco Systems, Inc.

However, a major drawback of telnet is that the client must know, before making any connection request, the Internet Protocol address ("IP address") of the firewall that is protecting the target network which the client is attempting to access. An IP address is a unique 32-bit binary number assigned to each firewall, router, host computer or other network element that communicates using IP. Obtaining the IP address of a firewall can be inconvenient or impractical because there are so many IP addresses currently assigned to network devices. Further, IP addresses normally are guarded closely by the network owner, because knowledge of an IP address enables unauthorized traffic to reach the device identified by the IP address.

Moreover, once a user successfully uses the authentication and authorization mechanism to secure a logical path through the firewall, the user may be restricted to one type of network traffic for the connection. For example, a firewall can be configured to provide a path through the firewall for a specific type of network traffic as specified by a user profile that is associated with each authenticated user. The user profile contains information on what the user is authorized to do on the network. The user profile may specify, for example, that the user may use only File Transfer Protocol ("FTP") traffic. Thus, the user may use the path through the firewall only for FTP traffic, for the duration of that connection. Furthermore, the user profile associated with the user contains a specific IP address that specifies the host or client from which the user can attempt to secure a logical path through the firewall. Thus, a user is not free to use any one of several computers that may be available to access the target network. Also, the user may not be free to use a client in a network that employs Dynamic Host Configuration Protocol (DHCP). DHCP assigns dynamic IP addresses to the devices on a network. Thus, a client in a DHCP environment can have a different IP address every time it connects to the network.

Based on the foregoing, there is a clear need for a mechanism allowing users to use remote access via the Internet without requiring advance knowledge of the IP address of the firewall router, and without restricting a user to a particular host or client.

In particular, there is a need for an authentication and authorization mechanism in the context of remote access via the Internet that does not rely on telnet and that allows the passage of different types of traffic for a given connection.

SUMMARY OF THE INVENTION

The foregoing needs, and other needs and objects that will become apparent for the following description, are achieved

3

in the present invention, which comprises, in one aspect, a method of controlling access of a client to a network resource using a network device that is logically interposed between the client and the network resource, the method comprising creating and storing client authorization information at the network device, wherein the client authorization information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource; receiving a request from the client to communicate with the network resource; determining, at the network device, whether the client is authorized to communicate with the network resource based on the authorization information; and reconfiguring the network device to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information.

One feature of this aspect is that creating and storing client authorization information comprises the steps of creating and storing in the network device a set of authorization information for each client that communicates with the network device.

According to another feature of this aspect is that creating and storing client authorization information comprises the steps of creating and storing in the network device an authentication cache for each client that communicates with the network device.

In another feature, creating and storing client authorization information comprises the steps of creating and storing in the network device a plurality of authentication caches, each authentication cache uniquely associated with one of a plurality of clients that communicate with the network device, each authentication cache comprising information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource.

According to still another feature, determining whether the client is authorized to communicate with the network resource comprises the step of determining whether information in the request identifying the client matches information in a filtering mechanism of the network device and the authorization information stored in the network device.

In another feature, determining whether the client is authorized to communicate with the network resource comprises the steps of: determining whether a source IP address of the client in the request matches information in a filtering mechanism of the network device; and if so, determining whether the source IP address matches the authorization information stored in the network device.

In another feature, determining whether the client is authorized to communicate with the network resource comprises the steps of: determining whether a source IP address of the client in the request matches information in a filtering mechanism of the network device; determining whether the source IP address matches the authorization information stored in the network device; and when the source IP address fails to match the authorization information stored in the network device, determining if user identifying information received from the client matches a profile associated with the user that is stored in an authentication server that is coupled to the network device.

In another feature, determining whether the client is authorized to communicate with the network resource comprises the steps of: determining whether client identifying

4

information in the request matches information in a filtering mechanism of the network device; determining whether the client identifying information matches the authorization information stored in the network device; and only when the client identifying information fails to match the authorization information stored in the network device, then: creating and storing new authorization information in the network device that is uniquely associated with the client; requesting login information from the client; authenticating the login information by communicating with an authentication server that is coupled to the network device; and updating the new authorization information based on information received from the authentication server.

According to another feature, requesting login information from the client comprises sending a Hypertext Markup Language login form to the client to solicit a username and a user password; and authenticating the login information by communicating with an authentication server that is coupled to the network device comprises determining, from a profile associated with a user of the client stored in the authentication server, whether the username and password are valid.

In another feature, the method further comprises the steps of: creating and storing an inactivity timer for each authentication cache, wherein the inactivity timer expires when no communications are directed from the client to the network resource through the network device during a predetermined period of time; removing the updated authentication information when the inactivity timer expires.

In another feature, determining whether the client is authorized to communicate with the network resource comprises the steps of: determining whether a source IP address in the request matches information in a filtering mechanism of the network device; determining whether the source IP address matches the authorization information stored in the network device; and only when the source IP address fails to match the authorization information stored in the network device, then: creating and storing in the network device a new authentication cache that is uniquely associated with the client; requesting login information from the client; authenticating the login information by communicating with an authentication server that is coupled to the network device; and updating the new authentication cache based on information received from the authentication server.

According to another feature, reconfiguring the network device comprises the steps of creating and storing one or more commands to the network device whereby one or more interfaces of the network device are modified to permit communications between the client and the network resource.

In another feature, the method further involves instructing the client to reload the network resource that was identified in the request from the client when it is determined that the client is authorized to communicate with the network resource.

According to another feature, the method further comprises the steps of waiting a pre-determined period of time, and instructing the client to reload the network resource that was identified in the request from the client when it is determined that the client is authorized to communicate with the network resource.

In another feature, the network device comprising a firewall that protects the network resource by selectively blocking messages initiated by client and directed to the network resource, the firewall comprising an external interface and an internal interface, the firewall comprising an Output Access Control List at the internal interface and an

5

Input Access Control List at the external interface, wherein reconfiguring the network device comprises the step of: substituting the IP address in a user profile information associated with a user of the client to create a new user profile information, wherein the user profile associated with the user of the client is received from an authentication server that is coupled to the network device; and adding the new user profile information as temporary entries to the Input Access Control List at the external interface and to the Output Access Control List at the internal interface.

According to still another feature, the method further involves: creating and storing an inactivity timer for the authorization information, wherein the inactivity timer expires when no communications are directed from the client to the network resource through the network device during a pre-determined period of time; associating the temporary entries with the authorization information and the client; and removing the temporary entries and the authorization information from the network device if the inactivity timer expires.

In another feature, the authorization information includes a table of hashed entries and wherein associating the temporary entries to the authorization information further comprises storing the temporary entries in the table of hashed entries.

In another feature, the network device comprising a firewall that protects the network resource by selectively blocking messages initiated by client and directed to the network resource, the firewall comprising an external interface and an internal interface, the firewall comprising an Output Access Control List at the external interface and an Input Access Control List at the internal interface, wherein reconfiguring the network device comprises the step of: substituting the IP address in a user profile information associated with a user of the client to create a new user profile information, wherein the user profile associated with the user of the client is received from an authentication server that is coupled to the network device; and adding the new user profile information as temporary entries to the Input Access Control List at the internal interface and to the Output Access Control List at the external interface.

In another feature, the method further involves: creating and storing an inactivity timer for the authorization information, wherein the inactivity timer expires when no communications are directed from the client to the network resource through the network device during a pre-determined period of time; associating the temporary entries with the authorization information and the client; and removing the temporary entries and the authorization information from the network device if the inactivity timer expires.

In another feature, the authorization information includes a table of hashed entries and wherein associating the temporary entries to the authorization information further comprises storing the temporary entries in the table of hashed entries.

According to another aspect, the invention encompasses computer system for controlling access of a client to a network resource using a network device that is logically interposed between the client and the network resource, comprising: one or more processors; a storage medium carrying one or more sequences of one or more instructions including instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of: creating and storing client authorization information at the network device, wherein the client

6

authorization information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource; receiving a request from the client to communicate with the network resource; determining, at the network device, whether the client is authorized to communicate with the network resource based on the authorization information; and reconfiguring the network device to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information.

According to another aspect, the invention involves a router that is logically interposed between a client and a network resource and that controls access of the client to the network resource, comprising: one or more processors; a storage medium carrying one or more sequences of one or more instructions including instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of: creating and storing client authorization information at the router, wherein the client authentication information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource; receiving a request from the client to communicate with the network resource; determining, at the router, whether the client is authorized to communicate with the network resource based on the authorization information; and reconfiguring the router to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information.

In other aspects, the invention encompasses a computer apparatus, a computer readable medium, and a carrier wave configured to carry out the foregoing steps.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

FIG. 1 is a block diagram that illustrates a computer system upon which an embodiment may be implemented;

FIG. 2 is a block diagram of a system providing an authentication proxy in a network environment;

FIG. 3 is a block diagram of the system in FIG. 2 showing certain internal details;

FIG. 4 is a block diagram of the system in FIG. 3 showing certain paths of network traffic;

FIG. 5A illustrates a display of a graphical user interface containing a dialog box for soliciting a username and password;

FIG. 5B illustrates a display of the graphical user interface informing of an authentication success;

FIG. 6 is a state diagram of states in which an authentication cache may execute;

FIG. 7A is a flow diagram of a process of proxy authentication;

FIG. 7B is a flow diagram of further steps in the process of FIG. 7A.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A method and apparatus for authentication and authorization proxy mechanisms for firewalls that protect networks

7

is described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

OPERATIONAL CONTEXT

The present invention may be implemented using various client protocols such as Telnet, File Transfer Protocol (FTP), or HyperText Transfer Protocol (HTTP). For purposes of illustration, the invention is described in the context of an HTTP client protocol.

In one embodiment, a user of a client that is part of a local area network ("LAN") attempts to remotely access a server ("target server") or some other resource, such as a peer client or device. The target server and or peer are part of a packet-switched private network that operates using TCP/IP and other Internet standards ("intranet"). The client is connected to the Internet through the LAN, and the intranet is also connected to the Internet. Alternatively, the client may be a stand-alone computer connected to the Internet through a dial-up connection or a digital communication service such as an Integrated Services Digital Network (ISDN) connection. In another embodiment, a user of a client from within the intranet attempts to access a target server or other resource that is not part of the same intranet as that of the client.

When the target server executes an HTTP server, the client can remotely access the target server over the Internet by using a Web browser to specify a Web page on the target server. Using a Web browser to specify a Web page is hereafter referred to as an "HTTP request" or as "transmitting HTTP packets." A Web page of the target server may be accessed using identifying information, such as a Uniform Resource Locator ("URL") and therefore the Web page is sometimes called the "target URL."

The HTTP packets are intercepted by a firewall that protects the intranet from unwanted network traffic originating from the Internet (inbound traffic) and can prevent users of clients from within the intranet from accessing undesirable web sites on the Internet (outbound traffic). For purposes of illustration, use of an embodiment with inbound traffic is described in further detail below.

Upon intercepting the HTTP packets, the firewall requests, from the client, authentication information such as username and password. In response to receiving the authentication information, the firewall performs an authentication and authorization process. If the username is successfully authenticated, then the firewall is dynamically configured to open a passageway for the HTTP packets as well as other types of network traffic initiated from the user on the client. The other types of network traffic that are permitted through the passageway are specified in a user profile for that particular user. In this context, "open a passageway" means that the firewall re-configures itself, in response to successful authentication, so that packets that would otherwise be barred are now allowed to pass.

In this configuration, the firewall provides an authentication and authorization mechanism that substitutes for an authentication and authorization mechanism elsewhere in the network. Accordingly, the mechanism described in this document is referred to as an "Authentication Proxy." The Authentication Proxy may comprise one or more software

8

components, executed by a router. In one embodiment, the Authentication Proxy can be enabled on a router interface to intercept traffic initiated from a client that is not yet authenticated. The Authentication Proxy is responsible for validating the user associated with the client and for applying the appropriate user profile to the router interface. The authentication and authorization process and the dynamic configuration of the firewall are described in further detail below.

FIG. 2 is a block diagram of a system 200 in which an embodiment of an Authentication Proxy can be used. Generally, system 200 includes a LAN 206, and a local, packet-switched network that uses Internet protocols, or intranet, 216. The LAN 206 and the intranet are both connected to a global network such as the Internet. The LAN 206 and intranet 216 are respectively located in logically distinct regions, such as first region 202 and second region 204, which may be geographically separate. A firewall router 210 is logically interposed between LAN 206 and the intranet 216.

LAN 206 is a local area network comprising any number of network devices 208a, 208b, 208c interconnected by one or more communications channels 209. Ethernet, Token Ring, other protocols can characterize the communications channels 209.

Firewall router 210 is a specialized router that carries out firewall functions. The firewall router 210 is coupled to intranet 216, and an authentication and authorization server 218 ("AAA server"). The firewall router 210 controls remote access to intranet 216. AAA server 218 is a computer, or a group of hardware or software components or processes that cooperate or execute in one or more computer systems. The AAA server 218 has access to a database 220 that stores authentication and authorization information on users ("user profile"). The firewall router 210 cooperates with the AAA server 218 to perform authentication and authorization services. In this way, firewall router 210 restricts and controls access of traffic originating from intranet 216 and directed to LAN 206.

Intranet 216 is one or more interconnected networks ("internetwork") each of which may comprise any number of network devices. A target server 222 is part of Intranet 216 and is a computer system that may be remotely accessed by a client on LAN 206. In certain embodiments, the AAA server 218 and database 220 may be coupled indirectly to firewall router 210 through the Internet. In this configuration, the AAA server 218 and database 220 are said to be one or more "hops" removed from the firewall router 210. A hop is the next place in the network to send a packet so that a packet will eventually reach its destination.

FIG. 3 is a block diagram showing certain internal details of the system in FIG. 2. In this example, one of the network devices 208a-208c of LAN 206 is a client 306, such as a personal computer or workstation. Client 306 is associated with a user 302. In certain embodiments, client 306 is configured with or coupled to multiple modems or ISDN bearer channels that can be used to establish one or more connections 308, 310 from client 306 to firewall router 210. In one embodiment, client 306 runs a browser 304, which is an application program used to retrieve and display Web pages or other information stored in target server 222. Commercial examples of browser 304 include Netscape Navigator® or Microsoft Internet Explorer®. User 302 can use browser 304 to send an HTTP request over LAN 206 and Intranet 216 to target server 222.

AUTHENTICATION AND AUTHORIZATION

FIG. 4 is a block diagram of the system of FIG. 3 that illustrates providing an Authentication Proxy.

FIG. 7A and FIG. 7B are flow diagrams that illustrate a method for carrying out proxy authentication at a router. As an example, the method of FIG. 7A and FIG. 7B is described below in the context of the system of FIG. 4. However, the method is not limited to this context.

As in FIG. 3, the system of FIG. 4 includes User 302 who is associated with Client 306. A Browser 304 is executed by Client 306, which is part of LAN 206. Client 206 communicates with firewall router 210 over LAN 206 using messages illustrated by paths 401, 403, 404, 408a, 408b. Firewall router 210 is coupled to client 306, to AAA Server 218, and to intranet 216. One function of Client 306 is to request and retrieve electronic documents, applications or services that are available at target server 222.

A filtering mechanism 219 is part of the configuration of Authentication Proxy 400. The filtering mechanism 219 contains information identifying one or more IP addresses of clients that are authenticated in the network to which the firewall 210 belongs.

Paths of network traffic are depicted by solid lines or dotted lines with arrowheads. Paths 401, 403, 404, 408a, 408b illustrate network traffic communicated between client 306 and firewall router 210. Paths 405 and 406 illustrate network traffic communicated between firewall router 210 and AAA server 218. Paths 409 and 410 illustrate network traffic communicated between the client 306 and target server 222. Each path represents one or more packets, messages or sessions communicated among the respective end elements. Paths 409 and 410 pass through firewall router 210 but do not stop within the firewall router.

Firewall router 210 has an external interface 420 and an internal interface 422. Each interface 420, 422 defines how firewall router 210 regulates the flow of traffic arriving at or sent from the respective interface. The external interface 420 includes an input Access Control List ("ACL") 424, and an output ACL 426. Internal interface 422 also includes an input ACL 428 and an output ACL 430.

Access control lists filter packets and can prevent certain packets from entering or exiting a network. Each ACL is a list of information that firewall router 210 may use to determine whether packets arriving at or sent from a particular interface may be communicated within or outside the firewall router. For example, in an embodiment, input ACL 424 may comprise a list of IP addresses and types of allowable client protocols. Assume that firewall router 210 receives an inbound packet from client 306 at external interface 420 that is intended for target server 222. If the IP address of client 306 is not stored in input ACL 424, then firewall router 210 will not forward the packet further within the circuitry or software of the firewall router. Output ACL 426 similarly controls the delivery of packets from firewall router 210 to resources located outside external interface 420. Input ACL 428 and output ACL 430 govern packet flow to or from internal interface 422.

The firewall router 210 also includes any number of authentication caches 432, 434. The access control lists are linked to the authentication caches. Each authentication cache represents a valid user authentication. Each authentication cache may include a table of hashed entries of information such as a source IP address, a destination IP address, a source port value, a destination port value, and state information.

Firewall router 210 also includes Authentication Proxy 400. In one embodiment, Authentication Proxy 400 is one or more software elements, modules or processes executed by firewall router 210 and coupled to the external interface 420,

internal interface 422, and authentication caches 432, 434. Authentication Proxy 400 may be configured to carry out the functions described in this document.

AUTHENTICATION

In an embodiment, Authentication Proxy 400 is activated or enabled at the firewall router 210. Authentication Proxy 400 may be implemented as a software process within firewall router 210 that may be accessed using commands in a standard command-line router programming language.

For purposes of illustrating an example of operation of authentication proxy 400, assume that Authentication Proxy 400 receives a request for something in a network that is protected by the Authentication Proxy, as shown by block 702 of FIG. 7A. For example, User 302 uses browser 304 to send an HTTP request from client 306 for an electronic document, application or resource available at target server 222. User 302 is not authenticated in intranet 216. The HTTP request travels along path 401 from client 306 to firewall router 210. Each packet of an HTTP request includes a header portion that contains one or more fields of information. The fields include, among other things, values for source IP address and destination IP address of that packet.

In block 704, packets of the request are examined. For example, when the HTTP request arrives at the external interface 420 of the firewall router 210, Authentication Proxy 400 examines packets of the request. In block 706, the process determines whether a source IP address of the request is found in the standard access control list. For example, Authentication Proxy 400 determines whether the source IP address in the header field of the packets corresponds to any entry in the filtering mechanism 219 configured in the Authentication Proxy 400.

If the test of block 706 is affirmative, then control passes to block 708 in which the authentication caches are searched for the source IP address. In block 710, the process tests whether the source IP address is found. For example, if Authentication Proxy 400 determines that the source IP address matches at least one IP address stored in the filtering mechanism 219, then the Authentication Proxy 400 attempts to authenticate the user 302. In the preferred embodiment, Authentication Proxy 400 searches authentication caches 432, 434 for the source IP address. The goal of this search is to determine if the source IP address of the HTTP packet corresponds to an entry in any of the authentication caches 432, 434.

Assume that the filtering mechanism 219 contains a source IP address that matches the IP address of client 306, so that the test of block 706 is affirmative. However, User 302 is not yet authenticated, so that the test of block 710 is negative. Thus, the authentication caches have no hashed entries that match the source IP address found in the header field of the HTTP packet. As a result, without further action the firewall router 210 will intercept and will not forward the HTTP packet, instead of providing it with a logical pass-way through the router.

In one embodiment, at this stage, control is passed to block 720 of FIG. 7B. Preferably, a new authentication cache 436 is created for User 302, as shown in block 720. Each authentication cache may operate in a plurality of states. FIG. 6 is a state diagram that illustrates the states in which an authentication cache may operate. Initially, the state of the new authentication cache 436 is set to the HTTP_Init state 602, as shown by block 722. The HTTP_Init state 602 indicates that the authentication and authori-

11

zation state for User 302 is at an initial state. In one embodiment, an HTTP packet is provided with a logical passageway through the firewall router 210 only when the state of an authentication cache that contains information

12

The login form may be composed using the HyperText Markup Language ("HTML") format. Table 1 presents source code for an appropriate HTML document that may be used for the login form of FIG. 5A.

TABLE 1

EXAMPLE LOGIN FORM

```

<HTML>
<HEAD>
<TITLE>
Authentication Proxy Login Page
</TITLE>
<HFAID>
<BODY BGCOLOR=#FFFFFF>
<H1><Cico "router name" Firewall<H1>
<H2>Authentication Proxy<H2><BR><P><P><P>
<FORM METHOD=POST ACTION="/">
<INPUT TYPE=HIDDEN NAME=TIMETAG VALUE=">
Username: <INPUT TYPE=TEXT NAME=UNAME><BR><BR>
Password: <INPUT TYPE=PASSWORD NAME=PWD><BR><BR>
<INPUT TYPE=SUBMIT NAME=SUBMIT VALUE=SUBMIT>
</FORM>
</BODY>
</HTML>

```

about User 302 is set to the HTTP_Estab state 606, which is described further below.

If the source IP address of the HTTP packet from client 306 does not match any of the entries in the filtering mechanism 219, then Authentication Proxy 400 denies passage to the HTTP packet and makes no attempt at authentication, as shown by block 707 of FIG. 7A. As a result, advantageously, the packet is turned away at the interface and never reaches internal software and hardware elements of the firewall router.

If Authentication Proxy 400 is not configured with the filtering mechanism 219, then Authentication Proxy 400 will intercept traffic, for purposes of authentication, from all hosts whose connection initiating packets arrive at a firewall interface for which Authentication Proxy 400 has been enabled. It is not practical or desirable to attempt authentication for every packet that arrives at the firewall. For example, there is no point in authenticating packets that arrive from sites known to be hostile or sites of an unknown nature. Thus, in one embodiment, a filtering mechanism is defined to enable the Authentication Proxy 400 to bypass traffic from such sites.

Referring again to FIG. 7B, after the new authentication cache is created, login information is requested from the client, as shown in block 724. For example, Authentication Proxy 400 obtains authentication information from User 302 by sending a login form to client 306. The login form is an electronic document that requests User 302 to enter username and password information, as shown by path 403.

FIG. 5A is an example of a graphical user interface ("GUI") representation of the login form 500 that may be sent by Authentication Proxy 400. The form 500 may be displayed by browser 304, as indicated by command area 502. Form 500 also includes a data entry pane 503 having a Username field 504, and a Password field 506. A User 302 may enter username information in Username field 504 and may enter password information in Password field 506. To communicate the username and password information to Authentication Proxy 400, the user selects a "Submit" button 508. In response, the login information is communicated to firewall router 210, as indicated by path 404.

Login information is received from the client, as shown by block 726. Block 726 may involve receiving username and password information in an HTTP message that is generated when a user fills in and submits the form of Table 1. In block 728, the user is authenticated using the login information and the AAA server 218. For example, upon receipt of the username and password information, Authentication Proxy 400 attempts to authenticate the user by sending the username and password to the AAA server 218, as shown by path 405. AAA server 218 has access to database 220, which contains user profiles of authorized users.

In block 730, the process tests whether authentication is successful. Successful authentication will occur when AAA server 218 verifies that the username and password information are recognized and correct. If Authentication Proxy 400 successfully authenticates User 302 using AAA Server 218, then AAA server 218 informs firewall router 210 of the successful authentication, as shown by path 406. In block 732, the process updates the current authentication cache with the source IP address of the client and with information contained in the user profile of User 302. In block 734, the firewall is re-configured to allow packets using protocols specified in user's profile of User 302 and associated with that IP address to pass through the firewall freely. Block 734 may involve creating and executing one or more router commands. In this way, subsequent authentication attempts against the authentication cache at the firewall will succeed, precluding the need to connect to AAA Server 218 repeatedly. This provides a significant advantage and improvement in authentication speed and efficiency.

Authentication Proxy 400 then informs Client 306 that authentication succeeded. As shown in block 736, the client is notified. In the preferred embodiment, Authentication Proxy 400 sends an HTML page containing a success message, as shown by path 408a.

FIG. 5B is an example of an HTML page that Authentication Proxy 400 may send to Client 306. Page 522 of window 520 contains an "Authentication Successful" message 524.

Page 522 also includes a second message 526 informing Client 306 that further action is being taken. For example,

13

the second message 526 may inform Client 306 that the resource originally requested from target server 222 is now being loaded or accessed. In the preferred embodiment, the original HTTP request is automatically forwarded by firewall router 210 to target server 222 without further intervention by Client 306 or User 302. Thus, Authentication Proxy 400 automatically interrupts and resumes the request.

As shown in block 738, the process waits. For example, after Authentication Proxy 400 sends the "Authentication Success" message 524 to User 302, the Authentication Proxy enters a wait state for a short, pre-determined period of time. During the wait state, a short period of time is allowed to elapse to enable client 306 and firewall router 210 to communicate handshaking messages and carry out related processing associated with establishing an HTTP connection. The delay period also allows the firewall router enough time to execute any commands that are issued as part of block 734. In one embodiment, a period of three (3) seconds elapses.

In block 740, the process sends a page reload instruction to the client. For example, after the delay period, Authentication Proxy 400 sends one or more messages to Client 306 that instruct the client to reload the target URL, as indicated by path 408b of FIG. 4. In response, Client 306 re-issues the original HTTP request for a resource on target server 222. By the time that the re-issued HTTP request arrives from Client 306 at firewall router 210, the firewall router has been re-configured to provide an logical path 409 circumscribed only by the user profile of User 302 for the connection to the target server using the process described further below. Similarly, return traffic may travel from target server 222 to Client 306 on path 410 subject to the authorization information of the user profile of User 302.

If the authentication is not successful, as shown in block 736 and block 738, the process may notify the client with an appropriate message or page, and block traffic from the source IP address. Alternatively, Authentication Proxy 400 may permit Client 306 to re-try authentication a predetermined number of times.

CREATING A PATH THROUGH THE FIREWALL USING DYNAMIC ACCESS CONTROL LISTS

When User 302 is authenticated successfully, Authentication Proxy 400 causes a user profile of User 302 to be downloaded from the AAA server 218 to the firewall router 210. Authentication Proxy 400 uses the user profile to dynamically configure the external interface 420 and internal interface 422 of firewall router 210 to provide a passageway for network traffic that initiates from User 302 on client 306. In this way, the types of traffic as specified by the user profile of User 302 and initiating from User 302 in the current session will pass freely through firewall router 210 to its destination.

In one embodiment, an authorized passageway through firewall router 210 is created using dynamic access control lists. In this embodiment, one or more entries may be added dynamically to each access control list 424, 426, 428, 430 of external interface 420 and internal interface 422.

When Authentication Proxy 400 sends a successful login confirmation message to User 302, a user profile of User 302 is downloaded from AAA server 218. In the preferred embodiment, the user profile is stored and retrieved in the form of one or more text commands, called proxy-access-list commands. Authentication Proxy 400 parses the proxy-access-list commands. Table 2 below contains samples of proxy-access-list commands. As each command is parsed,

14

Authentication Proxy 400 replaces the source IP address field of the command with the IP address of the client 306, to result in a modified proxy-access-list command. Each modified proxy-access-list command is added as a temporary entry to the access control lists at the external interface 420 and internal interface 422. Specifically, proxy-access-list commands are added as temporary entries to the input ACL 424 of the router's external interface 420 and to the output ACL 430 of the router's internal interface 422. Thus, access control lists 424, 426, 428, 430 will grow and shrink dynamically as entries are added and deleted.

When the temporary entries have been added to the ACL at the appropriate interfaces of the firewall router 210, the state of the current authentication cache is set to the HTTP...FINWAIT state 604. The HTTP...FINWAIT state 604 indicates that the authentication cache is associated with an authenticated user, but that its associated client is not yet connected to a target server.

TABLE 2

EXAMPLE PROXY-ACCESS-LIST COMMANDS

```

permit tcp host 192.168.25.215 any eq telnet
permit tcp host 192.168.25.215 any eq ftp
permit tcp host 192.168.25.215 any eq ftp-data
permit tcp host 192.168.25.215 any eq smtp
deny ip any any eq telnet
deny udp any any
permit ip any any (76 matches)
permit ip any any

```

AUTHENTICATION CACHE INACTIVITY TIMERS

Each authentication cache may have an inactivity timer. In the preferred embodiment, an inactivity cache is a software process that is maintained for each authentication cache based on the amount of traffic coming through firewall router 210 from Client 306. If the amount of traffic through the router for a particular client falls below a pre-determined threshold over a pre-determined period of time, then Authentication Proxy 400 generates an idle timeout event.

When an idle timeout event is generated for a particular client, Authentication Proxy 400 deletes the authentication cache associated with that client, and deletes all temporary entries in the access control lists that are associated with that particular client. This approach saves memory and ensures that the ACLs are regularly pruned.

For example, assume that the pre-determined inactivity timer value is 60 minutes. If Client 306 does not initiate any network traffic through the firewall router 210 for 60 minutes or more, then any subsequent traffic initiated from client 306 will be denied passage through the firewall router 210. In one embodiment, Authentication Proxy 400 will prompt User 302 to renew authentication for a new HTTP connection.

Preferably, the temporary entries in the ACLs are not automatically deleted when a user terminates a session. Instead, the temporary entries remain stored until the configured timeout is reached, or until they are specifically deleted by the system administrator. Using this process, the user is not required to log in a second time in the event of an inadvertent or transient disconnection, but unused entries are removed from the ACLs when a true idle condition occurs.

In addition, each temporary entry to an ACL 424, 426, 428, 430 is linked to an associated authentication cache 432,

15

434. Preferably, a temporary entry of an ACL is linked to an associated authentication cache by hashing the temporary entries and storing the hashed entries in the hash table that is maintained by the authentication cache.

CONNECTION AND COMMUNICATION WITH TARGET SERVER

When the interfaces of firewall router 210 have been configured with the new temporary entries in the ACLs, the result is that a logical passageway is opened through the firewall to allow certain types of traffic specified in the user profile of User 302 and initiating from Client 306 to pass unobstructed to target server 222. Using standard HTTP request-response communications, Client 306 establishes an HTTP connection to target server 222. When the connection to target server 222 is established, Authentication Proxy 400 changes the state of the authentication cache associated with the User 302 to the HTTP_ESTAB state 606.

As long as an idle timeout event does not occur, an authentication cache associated with a client remains valid. As long as the authentication cache associated with a client remains valid, the firewall router 210 does not attempt to intercept any subsequent traffic initiating from that client.

For example, assume that Client 306 continues to initiate network traffic or packets for passage through the firewall router 210 at least once every 60 minutes. Thus, the authentication cache 436 associated with client 306 remains valid and is not removed. Corresponding entries in the ACLs 424, 426, 428, 430 also remain valid. When a packet arrives at the firewall router 210, Authentication Proxy 400 determines that the source IP address of the packet matches an entry in the authentication cache 436. Accordingly, Authentication Proxy 400 will allow the packet to pass through firewall router 210.

Moreover, subsequent traffic initiating from client 306 may travel to any destination, as specified by the temporary entries to the dynamic ACLs that are associated with client 306. When a packet from client 306 arrives at firewall router 210, Authentication Proxy 400 checks the hashed entries in the authentication cache 436 for a match in the destination IP address. If a match is found, then the Authentication Proxy 400 will allow the packet to pass through firewall router 210 to the specified destination.

In addition, the temporary entries to the dynamic ACLs 424, 426, 428, 430 may specify various permissible communication protocols that client 306 may use. Thus, client 306 may initiate network traffic using any communication protocol specified in its associated temporary ACL entries, as long as an idle timeout event has not occurred. The client 306 is not restricted to TCP/IP protocol that was used to send the initial HTTP request. Examples of other communication protocols that client 306 may use include telnet, Internet Control Message Protocol ("ICMP"), and File Transfer Protocol ("FTP").

HARDWARE OVERVIEW

FIG. 1 is a block diagram that illustrates a computer system 100 upon which an embodiment of the invention may be implemented. In one embodiment, computer system 100 is a firewall device, such as a router.

Computer system 100 includes a bus 102 or other communication mechanism for communicating information, and a processor 104 coupled with bus 102 for processing information. Computer system 100 also includes a main memory 106, such as a random access memory (RAM) or other

16

dynamic storage device, coupled to bus 102 for storing information and instructions to be executed by processor 104. Main memory 106 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 104. Computer system 100 further includes a read only memory (ROM) 108 or other static storage device coupled to bus 102 for storing static information and instructions for processor 104. A storage device 110, such as non-volatile random-access memory (NVRAM), is provided and coupled to bus 102 for storing information and instructions.

Computer system 100 may be coupled via communication interface 117 to a terminal 112, such as a cathode ray tube (CRT) dumb terminal or workstation, for receiving command-line instructions from and displaying information to a computer user. Terminal 112 includes an input device such as a keyboard, and may include a cursor control such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 104.

Computer system 100 has a switching system 116 which provides a plurality of links or interfaces to a network. Switching system 116 provides a way to connect an incoming network link 114 to an outgoing network link 118. There may be many links 114, 118.

The invention is related to the use of computer system 100 for regulating packet traffic in an integrated services network. According to one embodiment of the invention, regulating packet traffic in an integrated services network is provided by computer system 100 in response to processor 104 executing one or more sequences of one or more instructions contained in main memory 106. Such instructions may be read into main memory 106 from another computer-readable medium, such as storage device 110. Execution of the sequences of instructions contained in main memory 106 causes processor 104 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor 104 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, NVRAM, such as storage device 110, or magnetic or optical disks. Volatile media includes dynamic memory, such as main memory 106. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 102. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor 104 for execution. For example, the instructions may initially be carried on a magnetic disk of a

remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 100 can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus 102. Bus 102 carries the data to main memory 106, from which processor 104 retrieves and executes the instructions. The instructions received by main memory 106 may optionally be stored on storage device 110 either before or after execution by processor 104.

Computer system 100 also includes a communication interface 117 coupled to bus 102. Communication interface 117 provides a two-way data communication coupling to a network link 120 that is connected to a local network 122. For example, communication interface 117 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 117 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 117 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 120 typically provides data communication through one or more networks to other data devices. For example, network link 120 may provide a connection through local network 122 to a host computer 124 or to data equipment operated by an Internet Service Provider (ISP) 126. ISP 126 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 128. Local network 122 and Internet 128 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 120 and through communication interface 117, which carry the digital data to and from computer system 100, are exemplary forms of carrier waves transporting the information.

Computer system 100 can send messages and receive data, including program code, through the network(s), network link 120 and communication interface 117. In the Internet example, a server 130 might transmit a requested code for an application program through Internet 128, ISP 126, local network 122 and communication interface 117. In accordance with the invention, one such downloaded application provides for regulating packet traffic in an integrated services network as described herein.

The received code may be executed by processor 104 as it is received, and/or stored in storage device 110, or other non-volatile storage for later execution. In this manner, computer system 100 may obtain application code in the form of a carrier wave.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

What is claimed is:

1. A method of controlling access of a client to a network resource, the method comprising the steps of:
creating and storing client authorization information at a network firewall routing device that is logically interposed between the client and the network resource,

wherein the client authorization information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client has with respect to the network resource;

receiving a request from the client to communicate with the network resource;

determining, at the network firewall routing device, whether the client is authorized to communicate with the network resource based on the authorization information; and

reconfiguring the network firewall routing device to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information.

2. A method as recited in claim 1, wherein creating and storing client authorization information comprises the steps of creating and storing in a cache in the network firewall routing device a set of authorization information for each client that communicates with the network routing device.

3. A method as recited in claim 1, wherein creating and storing client authorization information comprises the steps of creating and storing in the network firewall routing device an authentication cache for each client that communicates with the network firewall routing device.

4. A method as recited in claim 1, wherein creating and storing client authorization information comprises the steps of creating and storing in the network firewall routing device a plurality of authentication caches, each authentication cache uniquely associated with one of a plurality of clients that communicate with the network firewall routing device, each authentication cache comprising information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource.

5. A method as recited in claim 1, wherein determining whether the client is authorized to communicate with the network resource comprises the step of determining whether information in the request identifying the client matches information in a filtering mechanism of the network routing device and the authorization information stored in the network firewall routing device.

6. A method as recited in claim 1, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address of the client in a data packet of the request matches information in a filtering mechanism of the network firewall routing device; and

if so, determining whether the source IP address matches the authorization information stored in the network routing device.

7. A method as recited in claim 1, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address of the client in a data packet of the request matches information in a filtering mechanism of the network firewall routing device;

if a match is found using the filtering mechanism, determining whether the source IP address matches the authorization information stored in the network firewall routing device; and

when the source IP address fails to match the authorization information stored in the network firewall routing

19

device, determining if user identifying information received from the client matches a profile associated with the user that is stored in an authentication server that is coupled to the network firewall routing device.

8. A method as recited in claim 1, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether client identifying information in the request matches information in a filtering mechanism of the network firewall routing device;

if a match is found using the filtering mechanism determining whether the client identifying information matches the authorization information stored in the network firewall routing device; and

only when the client identifying information fails to match the authorization information stored in the network firewall routing device, then:

creating and storing new authorization information in the network firewall routing device that is uniquely associated with the client;

requesting login information from the client; authenticating the login information by communicating with an authentication server that is coupled to the network firewall routing device; and

updating the new authorization information based on information received from the authentication server.

9. A method as recited in claim 8, wherein:

requesting login information from the client comprise sending a Hypertext Markup Language login form from the network firewall routing device to the client to solicit a username and a user password; and

authenticating the login information by communicating with an authentication server that is coupled to the network firewall routing device comprises determining, from a profile associated with a user of the client stored in the authentication server, whether the username and password me valid.

10. A method as recited in claim 8, further comprising the steps of:

creating and storing an inactivity timer for each authentication cache, wherein the inactivity timer expires when no communications are directed from the client to the network resource through the network firewall routing device during a pre-determined period of time; removing the updated authentication information when the inactivity timer expires.

11. A method as recited in claim 1, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address in the request matches information in a filtering mechanism of the network firewall routing device;

determining whether the source IP address matches the authorization information stored in the network firewall routing device using an authentication cache in the network firewall routing device; and

only when the source IP address fails to match the authorization information stored in the network firewall routing device, then:

creating and storing a new entry in the authentication cache that is uniquely associated with the client; requesting login information from the client;

authenticating the login information by communicating with an authentication server that is coupled to the network firewall routing device; and

20

updating the new entry in the authentication cache based on information received from the authentication server.

12. A method as recited in claim 1, wherein reconfiguring the network firewall routing device comprises the steps of creating and storing one or more commands to the network firewall routing device which, when executed by the network firewall routing device, result in modifying one or more routing interfaces of the network firewall routing device to permit communications between the client and the network resource.

13. A method of controlling access of a client to a network resource using a network firewall routing device that is logically interposed between the client and the network resource, the method comprising the steps of:

creating and storing client authorization information at the network firewall routing device, wherein the client authorization information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client has with respect to the network resource;

receiving a request from the client to communicate with the network resource;

determining, at the network firewall device, whether the client is authorized to communicate with the network resource based on the authorization information; and

reconfiguring the network firewall routing device to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information;

the network firewall routing device comprising a firewall that protects the network resource by selectively blocking messages initiated by client and directed to the network resource, the firewall comprising an external interface and an internal interface, the firewall comprising an Output Access Control List at the internal interface and an Input Access Control List at the external interface, wherein reconfiguring the network firewall routing device comprises:

substituting the IP address in a user profile information associated with a user of the client to create a new user profile information, wherein the user profile associated with the user of the client is received from an authentication server that is coupled to the network firewall routing device; and

adding the new user profile information as temporary entries to the Input Access Control List at the external interface and to the Output Access Control List at the internal interface.

14. A method as recited in claim 13, further comprising the steps of:

creating and storing an inactivity timer for the authorization information, wherein the inactivity timer expires when no communications are directed from the client to the network resource through the network firewall routing device during a pre-determined period of time; associating the temporary entries with the authorization information and the client; and removing the temporary entries and the authorization information from the network firewall routing device if the inactivity timer expires.

15. A method as recited in claim 14, wherein the authorization information includes a table of hashed entries and

21

wherein associating the temporary entries to the authorization information further comprises storing the temporary entries in the table of hashed entries.

16. A method of controlling access of a client to a network resource using a network firewall routing device that is logically interposed between the client and the network resource, the method comprising the steps of:

creating and storing client authorization information at the network firewall routing device, wherein the client authorization information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client has with respect to the network resource;

receiving a request from the client to communicate with the network resource;

determining, at the network firewall routing device, whether the client is authorized to communicate with the network resource based on the authorization information; and

reconfiguring the network firewall routing device to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information;

the network firewall routing device comprising a firewall that protects the network resource by selectively blocking messages initiated by client and directed to the network resource, the firewall comprising an external interface and an internal interface, the firewall comprising an Output Access Control List at the external interface and an Input Access Control List at the internal interface, wherein reconfiguring the network firewall routing device comprises the step of:

substituting the IP address in a user profile information associated with a use of the client to create a new user profile information, wherein the user profile information associated with the user of the client is received from an authentication server that is coupled to the network firewall routing device; and

adding the new user profile information as temporary entries to the Input Access Control List at the internal interface and to the Output Access Control List at the external interface.

17. A method as recited in claim 16, further comprising the steps of:

creating and storing an inactivity timer for the authorization information, wherein

the inactivity timer expires when no communications are directed from the client to the network resource through the network firewall routing device during a pre-determined period of time;

associating the temporary entries with the authorization information and the client; and

removing the temporary entries and the authorization information from the network firewall routing device if the inactivity timer expires.

18. A method as recited in claim 17, wherein the authorization information includes a table of hashed entries and wherein associating the temporary entries to the authorization information further comprises storing the temporary entries in the table of hashed entries.

19. A computer-readable medium carrying one or more sequences of one or more instructions for controlling access of a client to a network resource using a network firewall routing device, the one or more sequences of one or more

22

instructions including instructions which, when executed by one or more processors, cause the one or more processors to perform the steps of:

creating and storing client authorization information at the network firewall routing device that is logically interposed between the client and the network resource, wherein the client authorization information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client has with respect to the network resource;

receiving a request from the client to communicate with the network resource;

determining, at the network firewall routing device, whether the client is authorized to communicate with the network resource based on the authorization information; and

reconfiguring the network firewall routing device to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information.

20. A computer system for controlling access of a client to a network resource using a network firewall routing device, comprising:

one or more processors;

a storage medium carrying one or more sequences of one or more instructions including instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of:

creating and storing client authorization information at the network firewall routing device that is logically interposed between the client and the network resource, wherein the client authorization information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client has with respect to the network resource;

receiving a request from the client to communicate with the network resource;

determining, at the network firewall routing device, whether the client is authorized to communicate with the network resource based on the authorization information; and

reconfiguring the network firewall routing device to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information.

21. A data packet firewall router that is logically interposed between a client and a network resource and that controls access of the client to the network resource, comprising:

one or more processors;

a storage medium carrying one or more sequences of one or more instructions including instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of:

creating and storing client authorization information at the router, wherein the client authentication information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access

23

privileges the client has with respect to the network resource;
receiving a request from the client to communicate with the network resource;
determining, at the router, whether the client is authorized to communicate with the network resource based on the authorization information; and

24

reconfiguring the router to permit the client to Communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,463,474 B1
DATED : October 8, 2002
INVENTOR(S) : Fuh et al.

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 18,

Line 57, replace "network comprises" with -- network resources comprises --.

Column 19,

Line 29, replace "comprise" with -- comprises --.

Line 38, replace "me" with -- are --.

Column 24,

Lines 1-2, replace "Communicate" with -- communicate --.

Signed and Sealed this

Twenty-fifth Day of March, 2003

A handwritten signature in black ink, appearing to read "James E. Rogan", written over a horizontal line.

JAMES E. ROGAN
Director of the United States Patent and Trademark Office



US006714979B1

(12) United States Patent
Brandt et al.**(10) Patent No.: US 6,714,979 B1**
(45) Date of Patent: Mar. 30, 2004**(54) DATA WAREHOUSING INFRASTRUCTURE FOR WEB BASED REPORTING TOOL****(75) Inventors:** **Andre R. Brandt**, Colorado Springs, CO (US); **Barbara Frueh**, Colorado Springs, CO (US); **Sajan J. Pillai**, Colorado Springs, CO (US); **Karl Rehder**, Colorado Springs, CO (US); **Donald J. Shearer**, Colorado Springs, CO (US)**(73) Assignee:** **WorldCom, Inc.**, Clinton, MS (US)**(*) Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.**(21) Appl. No.: 09/159,402****(22) Filed: Sep. 24, 1998****Related U.S. Application Data****(60)** Provisional application No. 60/060,655, filed on Sep. 26, 1997.**(51) Int. Cl.⁷** **G06F 15/173****(52) U.S. Cl.** **709/225; 709/205; 709/217; 709/223; 709/227; 709/229; 707/3; 707/4; 707/9; 707/10****(58) Field of Search** **709/205, 225, 709/227, 224, 219, 217, 228, 229, 223; 705/26, 18, 27, 51, 34; 713/155, 201, 152; 707/102, 523, 513, 2, 9, 3, 6, 10; 379/114.01, 114.02, 114.28, 114.29****(56) References Cited****U.S. PATENT DOCUMENTS**

4,160,129 A	7/1979	Peyser et al.	379/220.01
4,345,315 A	8/1982	Cadotte et al.	705/10
4,817,050 A	3/1989	Komatsu et al.	707/10
4,823,373 A	4/1989	Takahashi et al.	
4,893,248 A	1/1990	Pitts et al.	705/400
4,972,504 A	11/1990	Daniel, Jr. et al.	705/10
5,041,972 A	8/1991	Frost	705/10
5,075,771 A	12/1991	Hashimoto	725/46

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	0 809 387 A2	5/1997
JP	09064870 A	3/1997
WO	WO97/11443	3/1997
WO	WO 97/16911	5/1997
WO	WO 97/23988	7/1997
WO	WO 98/19472	5/1998
WO	WO 99/01826	1/1999
WO	00/11573	3/2000

OTHER PUBLICATIONS

Quadri et al., Hewlett-Packard and Cisco Systems, Internet Usage Platform White Paper.*

HP and Cisco Deliver Internet Usage Platform and Billing and Analysis Solutions, New Platform and Solutions Allow ISPs and Carriers to Offer Value-added Services.*

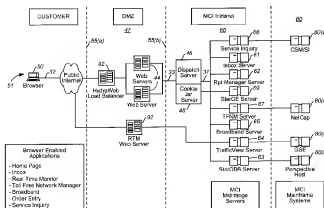
"HP Smart Internet, Transform User Data Into Revenue".* HP Smart Internet Usage Analysis Solution, Transform User Data Into Competitive Advantage.*

HP/Cisco, Internet Usage Platform, Transforming Internet Services Into Revenue.*

(List continued on next page.)

Primary Examiner—David Wiley*Assistant Examiner*—William C. Vaughn, Jr.**(57) ABSTRACT**

A data warehousing infrastructure for telecommunications priced call detail data is integrated with a Web/Internet based reporting system providing a common GUI enabling the requesting, customizing, scheduling and viewing of various types of priced call detail data reports. Such an infrastructure performs an extraction process to obtain only those billing detail records of entitled customers, and a harvesting process for transforming the billing records into a star schema format for storage in one or more operational data storage devices. The system is integrated with a database server supporting expedient and accurate access to the priced telecommunications priced call detail data for customer call detail data report generation.

37 Claims, 23 Drawing Sheets

US 6,714,979 B1

Page 2

U.S. PATENT DOCUMENTS

5,131,020 A	7/1992	Liebeshy et al.	455/422	5,781,632 A	7/1998	Odom	705/78
5,136,707 A	* 8/1992	Block et al.	707/201	5,787,160 A	7/1998	Chaney et al.	
5,223,699 A	* 6/1993	Flynn et al.	235/280	5,787,412 A	7/1998	Bosch et al.	707/2
5,228,076 A	7/1993	Hopner et al.	379/93.17	5,790,780 A	8/1998	Brichta et al.	
5,245,533 A	9/1993	Maishall	705/10	5,790,789 A	8/1998	Suarez	
5,262,760 A	11/1993	Iwamura et al.	345/861	5,790,797 A	8/1998	Shimada et al.	709/224
5,285,494 A	2/1994	Sprecher et al.	455/423	5,790,809 A	8/1998	Holmes	709/228
5,287,270 A	2/1994	Hardy et al.	705/34	5,793,762 A	8/1998	Penners et al.	
5,313,598 A	5/1994	Yamakawa	707/1	5,793,964 A	8/1998	Rogers et al.	
5,315,093 A	5/1994	Stewart	235/381	5,796,393 A	8/1998	MacNaughton et al.	
5,325,290 A	6/1994	Caulfield et al.	705/34	5,799,154 A	8/1998	Kuriyan	709/223
5,327,486 A	7/1994	Wolff et al.	379/93.23	5,802,320 A	9/1998	Biehr et al.	
5,361,259 A	11/1994	Hunt et al.	370/402	5,805,803 A	9/1998	Birtrell et al.	
5,369,571 A	11/1994	Metts	705/10	5,812,533 A	9/1998	Cox et al.	370/259
5,452,446 A	9/1995	Johnson	707/1	5,812,654 A	9/1998	Anderson et al.	
5,475,836 A	12/1995	Harris et al.		5,812,750 A	9/1998	Dev et al.	714/4
5,481,542 A	1/1996	Logston et al.	725/131	5,815,080 A	9/1998	Taguchi	
5,483,596 A	1/1996	Rosenow et al.		5,815,665 A	9/1998	Teper et al.	
5,490,060 A	2/1996	Malec et al.	705/10	5,819,225 A	10/1998	Eastwood et al.	
5,491,779 A	2/1996	Berjan		5,819,271 A	10/1998	Mahoney et al.	
5,506,893 A	* 4/1996	Buchet et al.	379/111	5,825,769 A	10/1998	O'Reilly et al.	370/360
5,526,257 A	6/1996	Lerner	705/10	5,825,890 A	10/1998	Digmal et al.	713/151
5,530,744 A	6/1996	Charalambous et al.		5,826,029 A	10/1998	Gong, Jr. et al.	
5,533,108 A	7/1996	Harris et al.	379/211.02	5,826,269 A	10/1998	Hussey	
5,537,611 A	7/1996	Rajagopal et al.	379/221.07	5,832,519 A	* 11/1998	Bowen et al.	707/201
5,539,734 A	7/1996	Durwell et al.	370/410	5,835,084 A	11/1998	Bailey et al.	
5,548,726 A	8/1996	Pettus		5,844,896 A	12/1998	Marks et al.	
5,551,025 A	8/1996	O'Reilly et al.	707/104.1	5,845,067 A	12/1998	Postei et al.	
5,555,290 A	* 9/1996	McLeod et al.	379/88.25	5,845,267 A	12/1998	Konen	
5,563,805 A	10/1996	Arbuckle et al.	709/204	5,848,233 A	12/1998	Radia et al.	713/201
5,566,351 A	10/1996	Crittenden et al.	710/4	5,848,396 A	12/1998	Gerace	705/10
5,586,260 A	12/1996	Hu		5,848,399 A	12/1998	Burke	705/27
5,602,918 A	2/1997	Chen et al.	713/153	5,850,517 A	12/1998	Vekler et al.	
5,610,915 A	3/1997	Elliott et al.	370/259	5,852,810 A	12/1998	Sotiroff et al.	
5,621,727 A	4/1997	Vandrenil		5,852,812 A	12/1998	Reeder	
5,623,601 A	4/1997	Vu		5,862,325 A	1/1999	Reed et al.	709/201
5,630,066 A	5/1997	Gosling	709/221	5,867,495 A	2/1999	Elliott et al.	370/352
5,649,182 A	7/1997	Reitz		5,870,558 A	2/1999	Branton, Jr. et al.	709/224
5,650,994 A	7/1997	Daley	370/259	5,875,236 A	* 2/1999	Jankowicz et al.	379/114
5,659,601 A	* 8/1997	Cheslog	379/121	5,877,759 A	3/1999	Bauer	709/317
5,666,481 A	9/1997	Lewis	714/4	5,881,232 A	3/1999	Schwaller et al.	709/224
5,671,354 A	9/1997	Bo et al.		5,883,948 A	* 3/1999	Dunn	379/114
5,689,645 A	11/1997	Schettler et al.		5,884,032 A	3/1999	Bustam et al.	709/204
5,692,130 A	11/1997	Icgljovic et al.		5,884,312 A	3/1999	Dustau et al.	707/10
5,692,181 A	11/1997	Anand et al.	707/102	5,892,900 A	4/1999	Ginter et al.	713/200
5,694,546 A	12/1997	Reisman	705/26	5,907,681 A	5/1999	Bates et al.	709/228
5,696,906 A	12/1997	Peters et al.	705/34	5,909,679 A	6/1999	Hall	707/4
5,699,403 A	12/1997	Rouineu		5,906,682 A	6/1999	Cowan et al.	707/9
5,699,528 A	12/1997	Hogan	705/40	5,915,001 A	6/1999	Uppaluru	379/88.22
5,706,502 A	1/1998	Foley et al.		5,920,542 A	7/1999	Henderson	370/217
5,708,780 A	1/1998	Levergood et al.		5,923,016 A	* 7/1999	Fredregill et al.	235/380
5,710,882 A	1/1998	Svennevik et al.	709/227	5,930,754 A	7/1999	Melichione et al.	705/10
5,721,908 A	2/1998	Lagarde et al.	707/10	5,930,804 A	7/1999	Yu et al.	370/104.1
5,721,913 A	2/1998	Ackroff et al.		5,933,142 A	8/1999	LaStrange et al.	345/788
5,727,129 A	3/1998	Barrett et al.		5,937,165 A	8/1999	Schwaller et al.	709/224
5,734,709 A	3/1998	DeWitt et al.		5,938,729 A	8/1999	Cote et al.	709/224
5,734,831 A	3/1998	Sanders		5,949,976 A	9/1999	Chappelle	709/224
5,742,762 A	4/1998	Scholl et al.		5,953,389 A	* 9/1999	Pruett et al.	370/9
5,742,763 A	4/1998	Jones	709/317	5,956,714 A	* 9/1999	Condon	707/201
5,742,768 A	4/1998	Gennaro et al.		5,958,016 A	* 9/1999	Chang et al.	709/202
5,742,905 A	4/1998	Pepe et al.		5,960,411 A	9/1999	Hartman et al.	705/26
5,745,754 A	4/1998	Lagarde et al.		5,961,602 A	9/1999	Thompson et al.	709/229
5,754,830 A	5/1998	Buiss et al.		5,963,925 A	10/1999	Kolling et al.	705/40
5,757,900 A	* 5/1998	Nagel et al.	379/207	5,966,695 A	* 10/1999	Melichione et al.	705/10
5,764,756 A	6/1998	Onweller		5,970,467 A	10/1999	Alavi	705/10
5,768,501 A	6/1998	Lewis		5,974,396 A	10/1999	Anderson et al.	705/10
5,774,660 A	6/1998	Brendel et al.		5,974,441 A	10/1999	Rogers et al.	709/200
5,778,178 A	7/1998	Arunachalam		5,982,864 A	11/1999	Jagdish et al.	379/120
5,778,377 A	7/1998	Marlin et al.	707/103	5,982,891 A	* 11/1999	Ginter et al.	380/4
5,781,550 A	7/1998	Templin et al.		5,983,350 A	11/1999	Minear et al.	713/201
				5,991,733 A	* 11/1999	Aleia et al.	705/8

5,991,746 A	11/1999	Wang	705/40
5,991,806 A	11/1999	McHann, Jr.	709/224
5,990,525 A	12/1999	Krishnaswamy et al.	370/352
5,999,965 A	12/1999	Kelly	709/202
5,999,972 A	12/1999	Gish	709/219
5,999,973 A	12/1999	Gilho et al.	709/223
6,003,079 A	12/1999	Friedrich et al.	709/224
6,006,265 A	12/1999	Rangan et al.	709/226
6,011,844 A	1/2000	Uppaluru et al.	379/220.01
6,012,090 A	1/2000	Chung et al.	709/219
6,014,647 A	1/2000	Nizari et al.	705/39
6,014,702 A	1/2000	King et al.	709/227
6,018,768 A	1/2000	Ullman et al.	709/218
6,021,409 A	2/2000	Burrows	707/102
6,023,762 A	2/2000	Dean et al.	713/193
6,029,182 A	2/2000	Nehab et al.	707/523
6,031,904 A	2/2000	An et al.	379/201.02
6,032,132 A	2/2000	Nelson	705/34
6,032,184 A	2/2000	Cogger et al.	709/223
6,041,325 A	3/2000	Shah et al.	707/10
6,041,357 A	3/2000	Kunzelman et al.	709/228
6,044,144 A	3/2000	Becker et al.	379/265.02
6,044,362 A	3/2000	Neely	705/34
6,049,602 A	4/2000	Foladare et al.	379/265.04
6,049,786 A	4/2000	Smorodinsky	705/59
6,052,450 A	4/2000	Allison et al.	379/127
6,058,170 A	5/2000	Jagadish et al.	379/119
6,058,381 A	5/2000	Nelson	705/40
6,064,667 A	5/2000	Gisby et al.	370/352
6,065,002 A	5/2000	Knotts et al.	707/4
6,065,059 A	5/2000	Shieh et al.	709/233
6,072,493 A	6/2000	Driskell et al.	345/854
6,073,105 A	6/2000	Sutcliffe et al.	705/1
6,073,122 A	6/2000	Wool	705/51
6,073,241 A	6/2000	Rosenberg et al.	713/201
6,078,891 A	6/2000	Riordan et al.	705/10
6,078,924 A	6/2000	Ainsbury et al.	707/101
6,084,953 A	7/2000	Bardehneuer et al.	379/114.01
6,085,171 A	7/2000	Leonard	705/26
6,085,190 A	7/2000	Sakata	707/6
6,088,451 A	7/2000	He et al.	380/255
6,088,796 A	7/2000	Gracovets et al.	713/152
6,091,808 A	7/2000	Wood et al.	379/201.04
6,094,655 A	8/2000	Rogens et al.	707/10
6,104,704 A	8/2000	Buhler et al.	370/252
6,105,131 A	8/2000	Carroll	713/155
6,108,700 A	8/2000	MacCabe et al.	709/224
6,108,782 A	8/2000	Fletcher et al.	713/153
6,112,238 A	8/2000	Boyd et al.	709/224
6,112,242 A	8/2000	Jois et al.	709/224
6,115,040 A	9/2000	Bladow et al.	345/335
6,115,458 A	9/2000	Taskett	379/114.2
6,115,693 A	9/2000	McDonough et al.	705/10
6,115,737 A	9/2000	Ely et al.	709/203
6,119,109 A	9/2000	Muratori et al.	705/400
6,122,258 A	9/2000	Brown	370/256
6,128,624 A	10/2000	Papierniak et al.	707/104.1
6,130,933 A	10/2000	Miloslavsky	379/90.01
6,131,095 A	10/2000	Low et al.	707/10
6,131,116 A	10/2000	Riggins et al.	709/219
6,134,584 A	10/2000	Chang et al.	709/219
6,137,869 A	10/2000	Voit et al.	379/114.01
6,145,001 A	11/2000	Scholl et al.	709/223
6,154,744 A	11/2000	Kenner et al.	707/10
6,161,102 A	12/2000	Yanagihara et al.	707/3
6,161,126 A	12/2000	Wies et al.	709/203
6,161,128 A	12/2000	Smyk	709/205
6,163,597 A	12/2000	Voit	
6,173,311 B1	1/2001	Hassett et al.	709/209
6,205,456 B1	3/2001	Nakao	707/531
6,212,506 B1	4/2001	Shah et al.	705/418

6,212,558 B1	4/2001	Antur et al.	709/221
6,240,450 B1	5/2001	Sharples et al.	709/224
6,182,113 B1	6/2001	Peterson et al.	
6,253,239 B1	6/2001	Shklar et al.	709/217
6,275,490 B1	8/2001	Mattiaro et al.	370/352
6,286,050 B1	9/2001	Pullen et al.	709/229
6,292,481 B1	9/2001	Voi et al.	370/352
6,295,551 B1	9/2001	Roberts et al.	709/205
6,337,858 B1	1/2002	Petty et al.	
6,377,993 B1	4/2002	Brandt et al.	709/227
2001/0003828 A1	1/2001	Nayanaswami	
2001/0001014 A1	5/2001	Atkins, III et al.	380/239

OTHER PUBLICATIONS

Release Note for Netflow FlowCollector Release 2.0, Copyright Jul. 1998 and Release Notes for Netflow FlowAnalyzer Release 1.0, Copyright Sep. 1997.*

HP Invent, Capturing the Usage Billing Advantage.*

Kenney, Kathken, "American Management Systems Launches Internet-Based Customer Care and Billing Tool for Telecom Firms", PR Newswire, New York, Oct. 9, 1996, extracted from <http://proquest.umi.com> on internet Feb. 28, 2002.

Morgan, Rick, "When Used Right, Internet can be Effective Marketing Tool", Madison Capital Times, Madison, WI, Nov. 8, 1996, extracted from <http://proquest.umi.com> on internet on Feb. 28, 2002.

Edwards, Morris, "The Electronic Commerce Juggernaut", Communication News, Nokomis, Sep. 1997, vol. 34, Issue 9, extracted from <http://proquest.umi.com> on Internet on Feb. 28, 2002.

"Cryptography and the Internet", www.echonyc.com/~ysue/crypt.html, 1995.

Lee et al., "Supporting Multi-User, Multi-Applet Workspaces in CBE", Computer Supported Cooperative Work 1996, Cambridge, MA.

"Netscape 2.0 Beta Hip or Hype?", www.plant.net.au/innovations/20beta.html, Planet Internet, 1995.

"Cryptography and the Internet", www.echonyc.com/~ysue/crypt.html, 1995.

Lee et al., "Supporting Multi-User, Multi-Applet Workspaces in CBE", Computer Supported Cooperative Work 1996, Cambridge, MA.

"Netscape 2.0 Beta Hip or Hype?", www.plant.net.au/innovations/20beta.html, Planet Internet, 1995.

Computer Networks, Andrew S. Tanenbaum, pp. 410-412. "XIIIR6.3 (Broadway) Overview", <http://www.x.org/broadway.htm>.

"Stac Unveils Windows NT 4.0 and Web Browser Support in New ReachOut 7", http://www.stac.com/news/pressrel/pr_ro7_unveil.html.

Biggs, M., "Help for the Web enhances customer support, reduces help desk load" *Infoworld*, Jun. 16, 1997, v. 19, No. 24, pp. 82+.

Burch, B., "AT&T, MCI to release new management tools", *Network World*, Jan. 17, 1994, p. 19.

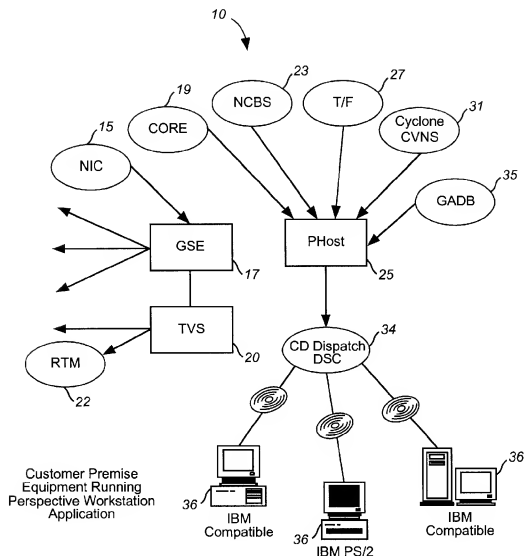
Low, C., "Integrating Communication Services", *IEEE Communication Magazine*, Jun. 1997, pp. 164-169.

"McAfee's New 'Self-Service' Help Desk Web Suite Makes PCs Help Desk-Ready", *Newswire Association Inc.*, Oct. 13, 1997.

Niemeyer, R., "Using Web Technologies in Two MLS Environments: A Security Analysis." *IEEE*, pp. 205-214, 1997.

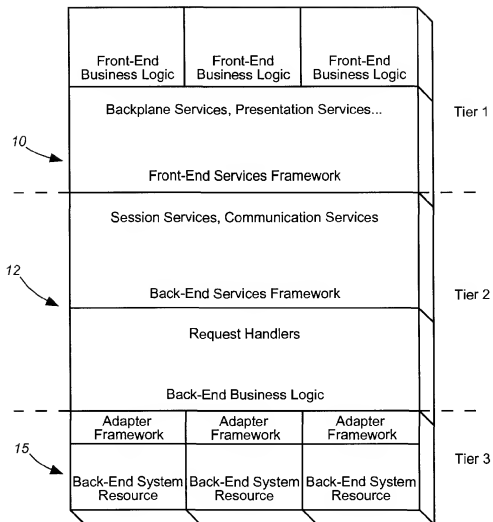
- Porter, T., "MCI offers tracking system: Direct Dispatch lets users eye problems remotely", *Service News*, Apr. 1994, p. 17.
- Shklar, L., et al., "MetaMagic: Generating Virtual Web Sites Through Data Modeling," <http://www.scope.gmd.de/info/www6/posters/714/poster714.html>.
- Vizard, M. et al., "MCI to Pilot Convergence Billing Service", *InfoWorld*, v. 18, Issue 37, Sep. 9, 1996.
- Yager, T., "Mixed Messages", *UNIX Review*, v. 16, n. 2, p. 29, Feb. 1998.
- "Carriers Improve Net Management Services", *Communications Week*, May 2, 1994, p. 74.
- "Network management; new software platform enhances network management capabilities; MCI ServiceView offers greater cost savings, increased flexibility", Product Announcement, *Edge*, Oct. 2, 1995, on & about AT&T, v. 10, n. 375, p. 11(1).
- "New software platform enhances network management capabilities . . .", *Business Wire*, Sep. 28, 1995 p. 9281122.
- "User's Guide: Microsoft Access", Microsoft Corporation, 1994, pp. 378,594,599, 630-632 (13).
- Jainschigg, J., "Billing confirmed: this easy-to-use box turns guest calls into revenue." *Teleconnect*, vol. 12, No. 9, p. 39(4).
- Chapman, D. Brent et al., "Building Internet Firewalls", Nov. 1995, O'Reilly & Associates, p. 58.
- He, Taniguchi, "Internet Traffic Control and Management Architecture", IEEE, Oct. 22-24, 1998, pp. s46-03-1-s46-03-5.
- Sixth International Conference on Network Protocols, IEEE, Technical Communication Services, Oct. 13-16, 1998, Table of Contents.
- Markovich, Robert, "WAN Service Level Management Could Keep Your Feet Out of the Fire, Ensure Carriers Diligence", *Network World*, Jul. 7, 1997.
- Meteorology; Databases, "Inforonics offers controlled access to Web Meteorology". *Information Today*, Apr. 1997, vol. 14 Issue 4, p53, 2p. This article reports that Inforonics has developed a controlled access gateway to MGA (Meteorological and Geostrophysical).
- Rosen,Michele,"BPCS steps into new millennium", Midrange Systems; Spring House; May 10, 1996. This article informs about the new release of BPCS Client/Server Software as the most extensive upgrade of the product since 1980s. It incorporates onject tech.
- Anonymous, "Call Accounting Products", *Teleconnect*, vol. 15, No. 3, p. 89, Mar. 1997.
- Deixler, Lyle, "Call Accounting Update", *Teleconnect*, vol. 15, No. 10, p. 87, Oct. 1997.
- Deixler, Lyle, "Micro-Tel's Microcall for Windows 95/NT", *Teleconnect*, vol. 15, No. 12, p. 35, Dec. 1997.
- Help-Desk Market Seeks Suite Success, *Computer Reseller News*, Jan. 5, 1998, p. 49.

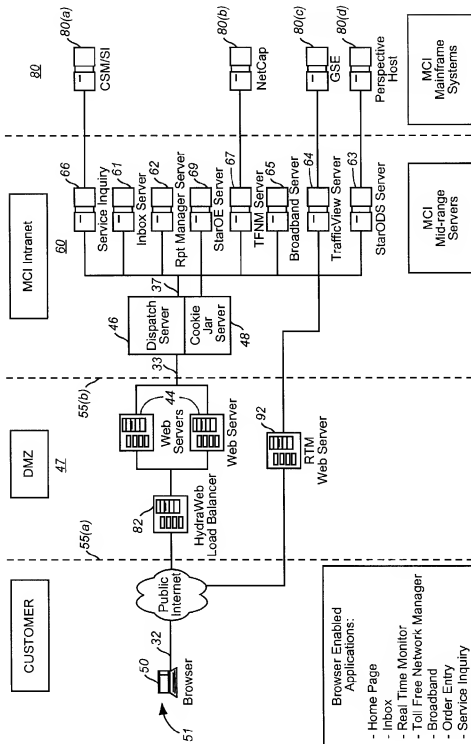
* cited by examiner



Prior Art

FIG. 1

**FIG. 2**



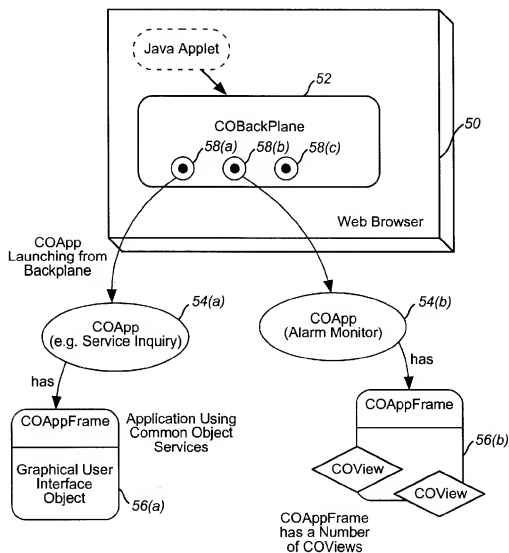


FIG. 4

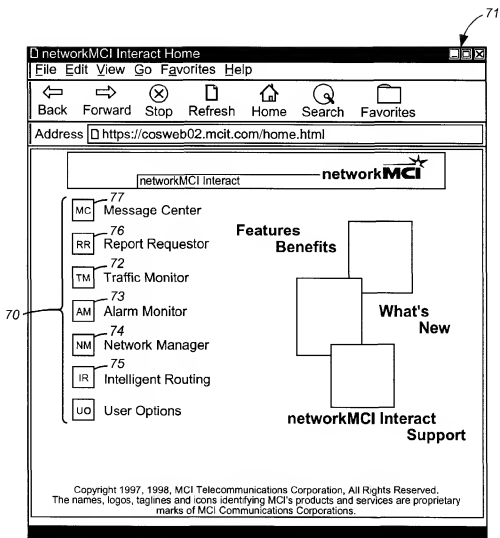


FIG. 5

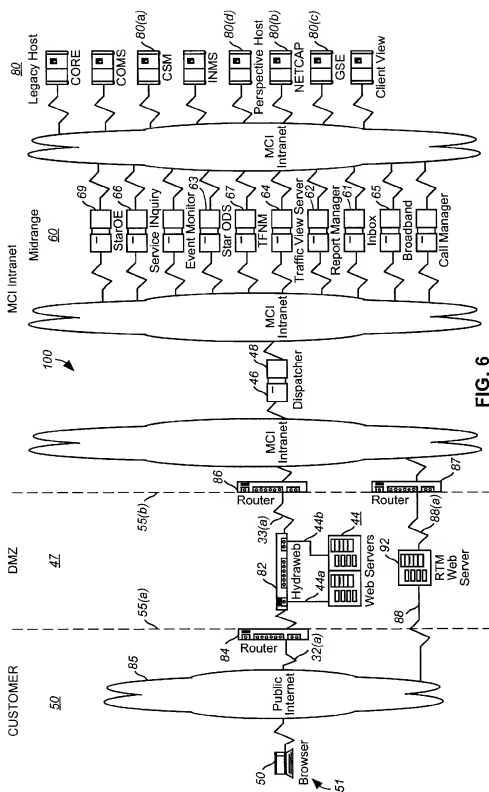


FIG. 6

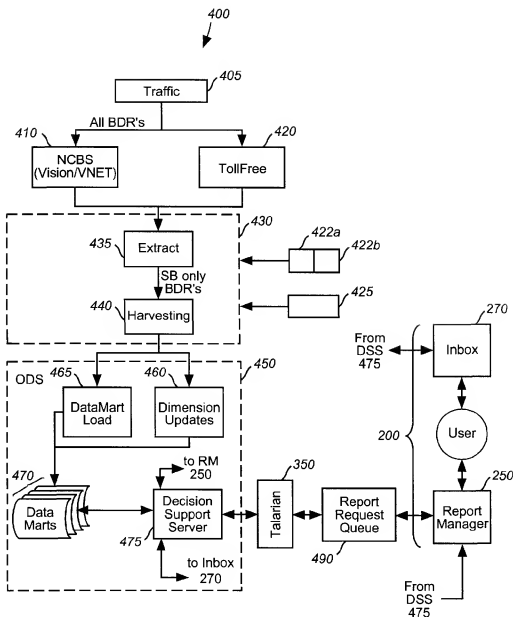


FIG. 7



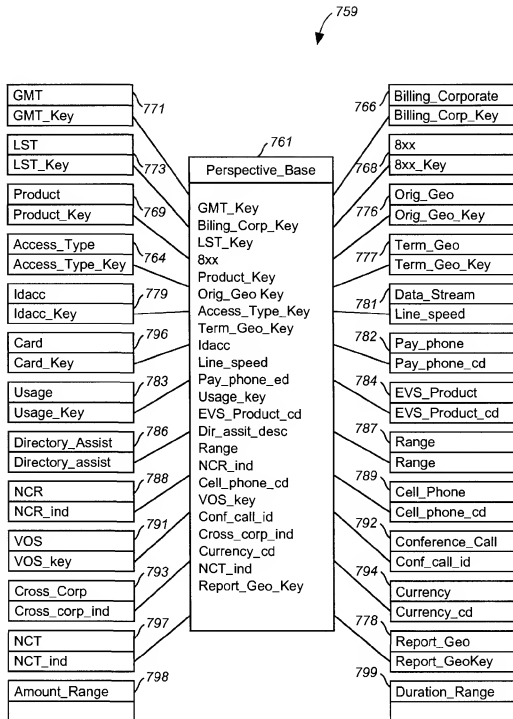


FIG. 9

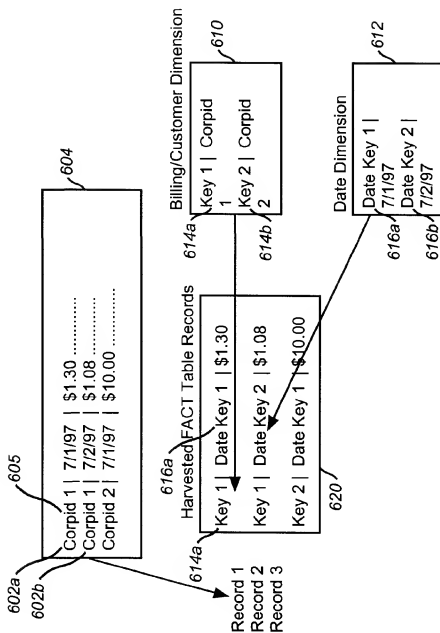


FIG. 10

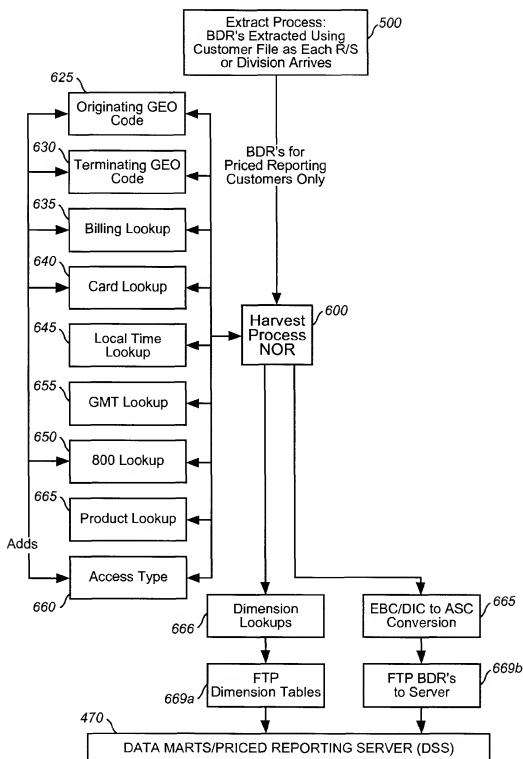


FIG. 11

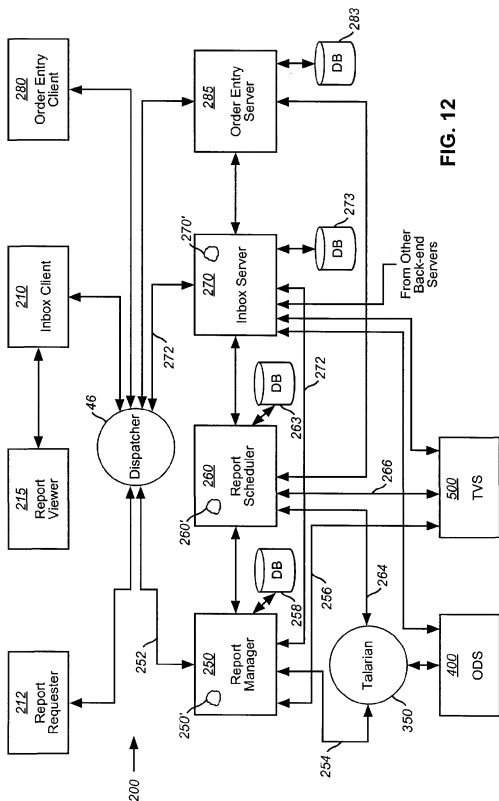


FIG. 12

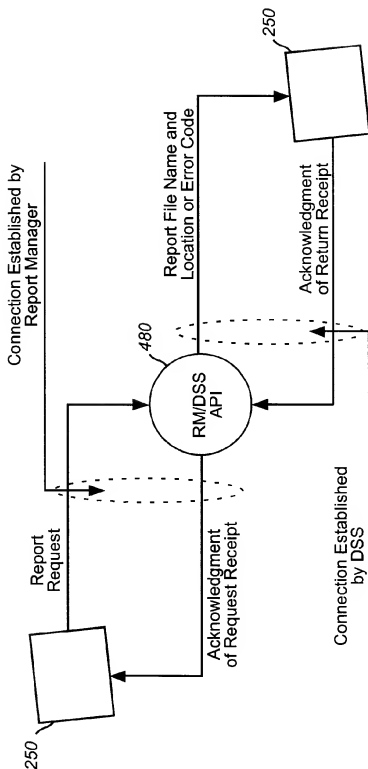


FIG. 13(a)

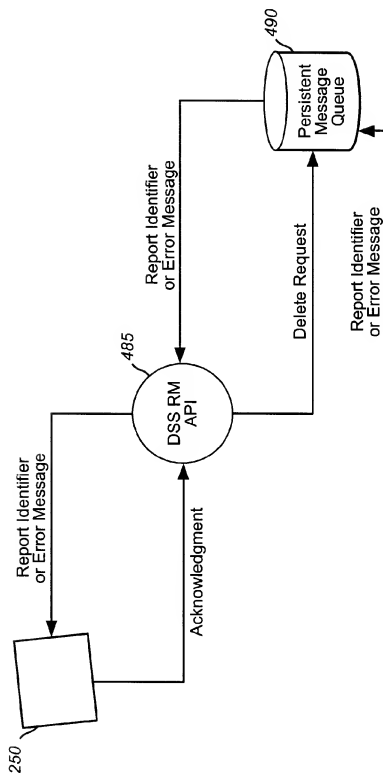
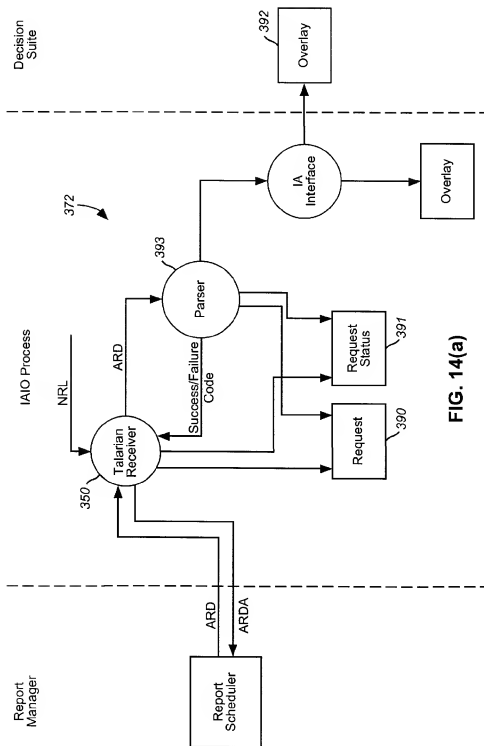
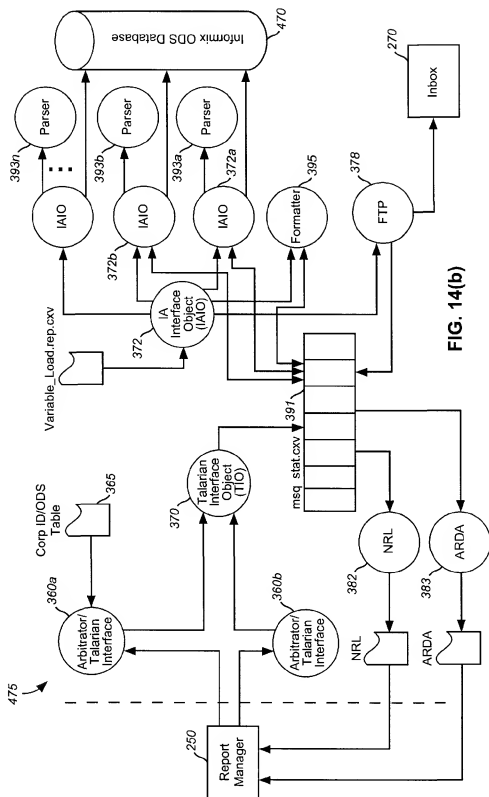


FIG. 13(b)





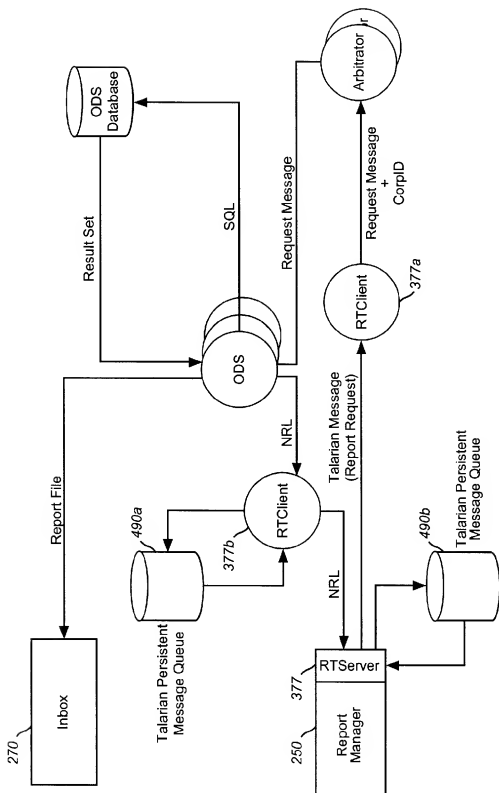


FIG. 15(a)

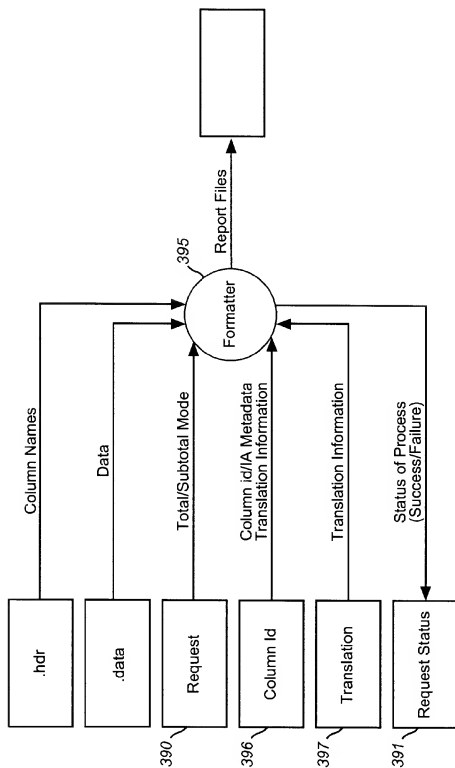
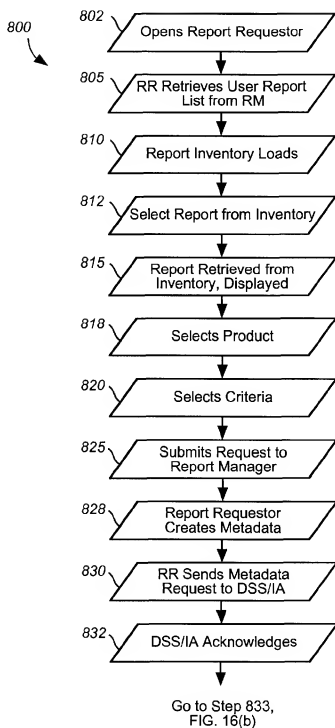


FIG. 15(b)

**FIG. 16(a)**

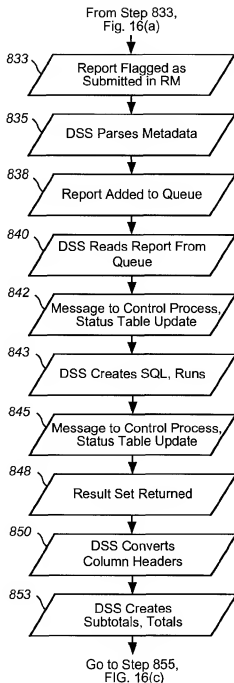


FIG. 16(b)

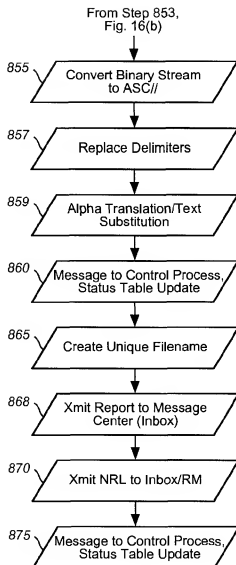


FIG. 16(c)

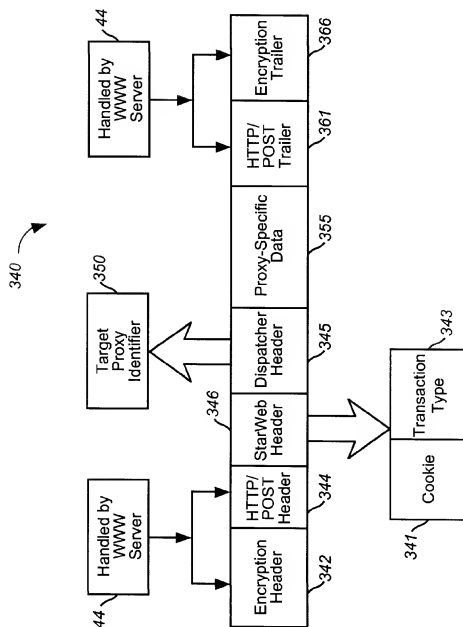


FIG. 17

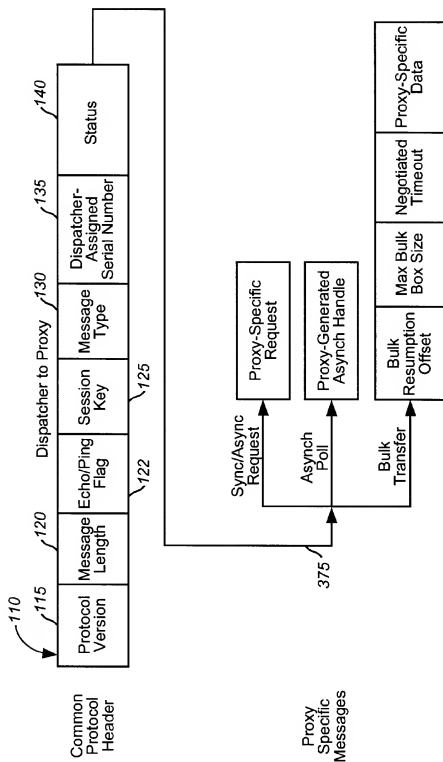


FIG. 18(a)

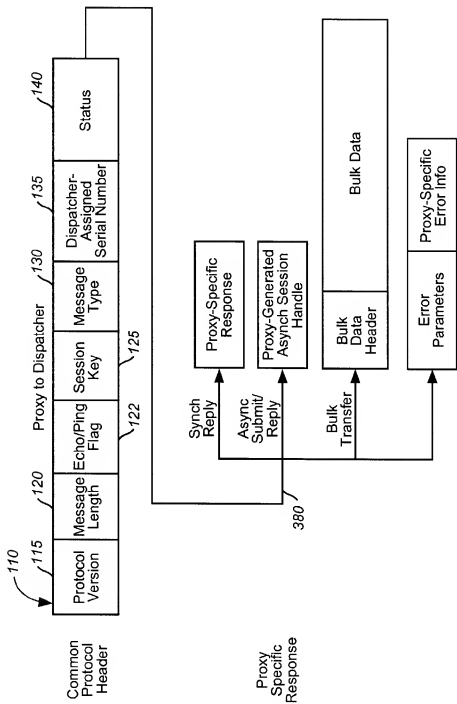


FIG. 18(b)

DATA WAREHOUSING INFRASTRUCTURE FOR WEB BASED REPORTING TOOL

CROSS-REFERENCE TO RELATED APPLICATIONS

The following patent application is based on and claims the benefit of U.S. Provisional Patent Application Serial No. 60/060,655, filed Sep. 26, 1997. This application is also related to the following applications which were all filed on Sep. 24, 1998. The related applications are U.S. application Ser. No. 09/159,695 (allowed); U.S. application Ser. No. 09/159,515 (issued Sep. 5, 2000 as U.S. Pat. No. 6,115,040); U.S. application Ser. No. 09/159,512; U.S. application Ser. No. 09/159,408; U.S. application Ser. No. 09/159,514; U.S. application Ser. No. 09/159,684; U.S. application Ser. No. 09/159,516; and U.S. application Ser. No. 09/159,409.

FIELD OF THE INVENTION

The present invention relates generally to information delivery systems and, particularly, to a novel, World Wide Web/Internet-based, telecommunications network data management reporting and presentation service for customers of telecommunications service entities.

BACKGROUND OF THE INVENTION

Telecommunications service entities, e.g., MCI, AT&T, Sprint, and the like, presently provide for the presentation and dissemination of customer account and network data management information to their customers predominantly by enabling customers (clients) to directly dial-up, e.g., via a modem, to the entity's application servers to access their account information, or, alternatively, via dedicated communication lines, e.g., ISDN, T-1, etc., enabling account information requests to be initiated through their computer terminal running, for example, a Windows®-based graphical user interface. The requests are processed by the entity's application servers, which retrieves the requested customer information, e.g., from one or more databases, processes and formats the information for downloading to the client's computer terminal.

Some types of data, e.g., "priced" call detail data pertaining to a customer's telecommunications number usage, is made available for customers in an aggregated or processed form and provided to customers, e.g., on a monthly basis. This type of data is analyzed to determine, for example, asset usage and trend information necessary, which is required for network managers to make critical business decisions. As an example, the assignee telecommunications carrier MCI Corporation provides an MCI ServiceView ("MSV") product line for its business customers which includes several client-server based data management applications. One of these applications, referred to as "Perspective," provides call usage and analysis information that focuses on the presentation of and priced call detail data and reports from an MCI Perspective Host Server ("PHost"). Another client-server based data management application, referred to as "Traffic View," focuses on the presentation of real time call detail data and network traffic analysis/monitor information as provided from an MCI Traffic view server. Particularly, with respect to MCI's Perspective system, customers are provided with their monthly priced and discounted raw call detail data, call detail aggregates, and statistical historical summary data. As such, the Perspective architecture is organized primarily as a batch midrange-based server data delivery mechanism with the data being typically delivered on a monthly basis, allowing for "delayed" trending, call

pattern analysis, repricing and invoice validation based on the customer's call detail data. The trending, analysis, and repricing functionality is maintained in workstation-based software provided to customers for installation at customer sites on their PCs.

FIG. 1 illustrates the current architecture 10 for Perspective and Traffic View Systems which presently run on separate environments and are maintained independently of each other. The StarPR server provides a batch reporting mechanism focused primarily on providing a billing data to 1-800/8xx, VNET, Vision, and other MCI customers and is used by MCI customers predominantly to do internal charge backs and to analyze billing usage. Alternately, or in addition, the customers use the data provided to them to do call traffic analysis, similar to TVS.

With specific reference to FIG. 1, the data collected is in the form of call detail records which are created by various MCI/Concert switches (not shown) whenever a telephone call is attempted in the MCI network and which includes information about call type, call origination and termination locations, date and time, added intelligent network services, any hop information, product type and other relevant information about the call. The Network Information Concentrator ("NIC") component 15 is a network element that collects the CDRs and sends them to appropriate locations via a Global Statistical Engine 17. The Global Statistical Engine 17 collects the CDRs and transforms, processes, and sends them to the TVS 20. The TVS provides access to this data through various statistical reports and real time monitoring engine 22 ("RIM").

The CDRs are also sent to the billing system which applied billing based on call detail values. These "priced" CDRs are known as Billing Detail Records ("BDRs") and are sent to a Perspective Host ("PHost") server 25. The PHost server 25 filters out the BDRs not pertaining to the "Perspective" customers, applies various transformations to the customer's raw call detail data to generate summary data, and generates and formats the data for the various Perspective customers. This data is then compressed, sent to a document service center ("DSC") and CD-ROM dispatcher ("CDD") 34 entities which respectively, uncompresses the data and burns CD-ROMs comprising the customer's raw call detail data and summary data, in addition to reference files and possibly application software (if not previously owned) enabling customers to perform analysis and trending of their Perspective data. These CD-ROMs are sent to the customers, usually on a billing cycle or monthly basis, who view their data through a Perspective workstation-based software application residing on that customer's CPE, e.g., PC or workstation 36.

As shown in FIG. 1, the existing Perspective Host 25 mainframe-based data delivery system interfaces with all Perspective upstream feed systems, including billing systems and order entry, and processes the data, e.g., creates canned aggregates, for delivery to the document service center.

The following upstream feed systems include: 1) order entry information from a customer order entry system 19 ("CORE") and which information is used by the Perspective Host to determine what customer data to process and where to send it; 2) VNET and Vision monthly billing data feeds from a commercial billing system ("NBCS") system 23; 3) a Tollfree monthly billing data feed from a T/F database feed 27; and, a Concert Virtual Network Services ("CVNS") product feed from a CVNS database 31. In order for all the CDR and data feed information to be processed by the PHost

3

server 25, various reference files and processing rules are provided including: alphanumeric translation reference files from the NCBS billing system 23 and an NP/XXX-state-city and country code lookup reference file originating from a calling area data base ("CADB") 35.

While effective for its purpose, the current data management and data storage legacy platform infrastructure only provides customers with their priced call detail data on a monthly basis, usually in the form of a canned report. This is not sufficient for an increasing number of customers who, to remain competitive, are required to have updated daily access to their data to enable them to make their critical business decisions quicker. Moreover, these legacy platforms including reporting data are reaching the architectural limits of scalability in terms of the total customers they can support, total online data they can present, total historical data they can keep and type and number of applications they can support.

Thus, a data warehousing approach that would support the daily processing and storage of customer's priced call detail data in a form suitable for expedient access and presentation as a report for customers over the World Wide Web/Internet would be highly desirable.

SUMMARY OF THE INVENTION

The present invention is a telecommunications data management/system architecture integrated with a novel Web/Internet based reporting system. Referred to as networkMCI Interact ("nMCI Interact"), the data management/system architecture implements a data warehouse approach to maintaining data obtained from upstream billing systems, i.e., priced call detail data, and which data may be made readily available for reporting on a daily basis. In this approach, priced call detail data is maintained in data marts, i.e., operational data stores, capable of meeting real-time processing and storage requirements. Particularly, these data marts may be partitioned based on various criteria, e.g., customer id, to enable easier management of data by providing scalability, and enabling more control of over hardware and software resources, in a cost-effective way. Additionally, this data mart approach may implement a back-end server component that receives data access requests from various users in the form of a report request, interactive data analysis request or data mining request. This server routes the query to the appropriate data marts, data warehouse or operational data store and responds to the requester with the result set.

The nMCI Interact Reporting system and data warehousing infrastructure is a layer functioning to enable customers to request reporting functionality across the Web/Internet. This report request functionality includes routing requests to appropriate data marts, e.g., real-time reporting requests may be satisfied by real-time database. Additionally, the interface provides customers with the ability to schedule and prioritize reports, format report request result sets, and provides for load balancing, report request validation, query generation and execution. Through a common GUI, customers are enabled to access their own billing call detail data.

In accordance with the principles of the invention, there is provided a Web/Internet based reporting system for providing timely delivery of a customer's priced telecommunications call detail data to a client workstation running a web browser application, the system comprising: a data warehousing infrastructure including: a process for generating a current customer list on a daily basis comprising customers entitled to receive daily telecommunications call

4

detail data; a device for receiving customer's raw telecommunications call detail data records from one or more telecommunications network switch mechanisms, and extracting certain call detail records for predetermined customers; a device for receiving the extracted call detail data records and replacing a call detail data item therein with a corresponding dimension key found in an associated dimension build table for that call detail item; and, a device for generating an output fact table comprising customer records having the unique key structures for enabling consolidated storage of specific customer call detail data; at least one secure server for managing client sessions over the Internet, the secure server supporting secure communication of customer request messages between the browser application client and the secure server; and, a device for receiving the customer requests from the secure server and generating corresponding database queries implementing the dimension keys for application against the output fact table to obtain a specific call customer's call detail data, the accessed call detail data being transmitted back to the client web browser via the secure server; whereby expedient and updated web/Internet-based access to the customer's daily call detail data is assured.

Advantageously, the data warehousing infrastructure for the Web/Internet based reporting system provides incremental, daily updates to data and allows users to report on either daily or monthly data. BDR's may be processed and loaded into data marts on a daily basis. On a monthly basis this daily data is replaced with records produced in the monthly billing run, after an audit approval. The goal of this replace operation is to replace daily priced records with the actual records used to compute monthly billing. This is necessary because a large percentage of the daily records, priced on a daily basis at tariff, may be repriced based on monthly discounts. Additionally, monthly data may also be updated based on audit anomalies. The monthly replace operation thus ensures the accuracy of the data used by customers.

Furthermore, the data management system permits use of existing hardware while allowing future growth to utilize new equipment at less cost and further, allows for incremental expansion as applications and database capacities grow.

BRIEF DESCRIPTION OF THE DRAWINGS

Further features and advantages of the invention will become more readily apparent from a consideration of the following detailed description set forth with reference to the accompanying drawings, which specify and show preferred embodiments of the invention, wherein like elements are designated by identical references throughout the drawings; and in which:

FIG. 1 illustrates conceptually an existing mainframe-based data delivery system 10 providing customer's call detail data;

FIG. 2 illustrates the software architecture component comprising a three-tiered structure;

FIG. 3 is a diagrammatic overview of the software architecture of the networkMCI Interact system;

FIG. 4 is an illustrative example of a backplane architecture schematic;

FIG. 5 illustrates an example client GUI presented to the client/customer as a browser web page;

FIG. 6 is a diagram depicting the physical networkMCI Interact system architecture;

5

FIG. 7 illustrates the primary data warehousing components implemented in the StarODS priced reporting component 400;

FIG. 8 illustrates the StarODS data extract process 500;

FIG. 9 illustrates the data model implemented for accessing information used in priced reporting system of nMCI Interact;

FIG. 10 depicts generally the creation of the BDR record fact table during the harvest process;

FIG. 11 illustrates the StarODS data harvest process 600;

FIG. 12 is a block diagram depicting the physical architecture of the StarWRS component of networkMCI Interact Reporting system;

FIG. 13(a) illustrates the logical Report Manager/DSS application programming interface;

FIG. 13(b) illustrates the logical DSS/Report Manager application programming interface;

FIGS. 14(a)–14(b) illustrate an overview of the process performed by the DSS in routing a request;

FIG. 15(a) illustrates an overview of the DSS connections enabling guaranteed message delivery in the nMCI Interact System;

FIG. 15(b) illustrates the formatter process implemented in the DSS server;

FIGS. 16(a)–16(c) illustrate the end-to-end process 800 for fulfilling priced report request;

FIG. 17 illustrates a logical message format sent from the client browser to the desired middle tier server for a particular application; and

FIGS. 18(a) and 18(b) are schematic illustrations showing the message format passed between the Dispatcher server and the application specific proxy (FIG. 18(a)) and the message format passed between the application specific proxy back to the Dispatcher server (FIG. 18(b)).

DETAILED DESCRIPTION OF THE INVENTION

The present invention is one component of an integrated suite of customer network management and report applications using a Web browser paradigm. Known as the networkMCI Interact system ("nMCI Interact") such an integrated suite of Web-based applications provides an invaluable tool for enabling customers to manage their telecommunication assets, quickly and securely, from anywhere in the world.

As described in co-pending U.S. patent application Ser. No. 09/159,695, filed Sep. 24, 1998 (D#11038), the nMCI Interact system architecture is basically organized as a set of common components comprising the following:

- 1) an object-oriented software architecture detailing the client and server based aspect of nMCI Interact;
- 2) a network architecture defining the physical network needed to satisfy the security and data volume requirements of the networkMCI System;
- 3) a data architecture detailing the application, back-end or legacy data sources available for networkMCI Interact; and
- 4) an infrastructure covering security, order entry, fulfillment, billing, self-monitoring, metrics and support.

Each of these common component areas will be generally discussed hereinbelow. A detailed description of each of these components can be found in a related, co-pending U.S.

6

patent application Ser. No. 09/159,695, filed Sep. 24, 1998 entitled INTEGRATED CUSTOMER INTERFACE SYSTEM FOR COMMUNICATIONS NETWORK MANAGEMENT, the disclosure of which is incorporated herein by reference thereto.

FIG. 2 is a diagrammatic illustration of the software architecture component in which the present invention functions. A first or client tier 40 of software services are resident on a customer work station and provides customer access to the enterprise system, having one or more downloadable application objects directed to front end business logic, one or more backplane service objects for managing sessions, one or more presentation services objects for the presentation of customer options and customer requested data in a browser recognizable format and a customer supplied browser for presentation of customer options and data to the customer and for Internet communications over the public Internet. Additionally applications are directed to front end services such as the presentation of data in the form of tables and charts, and data processing functions such as sorting and summarizing in a manner such that multiple programs are combined in a unified application suite. A second or middle tier 42, is provided having secure web servers and back end services to provide applications that establish user sessions, govern user authentication and their entitlements, and communicate with adaptor programs to simplify the interchange of data across the network.

A third or back end tier 45 having applications directed to legacy back end services including database storage and retrieval systems and one or more database servers for accessing system resources from one or more legacy hosts.

Generally, as explained in commonly owned, co-pending U.S. patent application Ser. No. 09/159,515, filed Sep. 24, 1998 (issued as U.S. Pat. No. 6,115,040 on Sep. 5, 2000), entitled GRAPHICAL USER INTERFACE FOR WEB ENABLED APPLICATIONS, the disclosure of which is incorporated herein by reference thereto, the customer workstation includes client software capable of providing a platform-independent, browser-based, consistent user interface implementing objects programmed to provide a reusable and common GUI abstraction and problem-domain abstractions. More specifically, the client software is created and distributed as a set of Java classes including the applet classes to provide an industrial strength, object-oriented environment over the Internet. Application-specific classes are designed to support the functionality and server interfaces for each application with the functionality delivered through the system being of two-types: 1) cross-product, for example, inbox and reporting functions, and 2) product specific, for example, toll free network management or Call Manager functions. The system is capable of delivering to customers the functionality appropriate to their product mix.

FIG. 3 is a diagrammatic overview of the software architecture of the networkMCI Interact system including: the Customer Browser (a.k.a. the Client) 50; the Demilitarized Zone (DMZ) 47 comprising a Web Servers cluster 44; the MCI Intranet Dispatcher Server 46; and the MCI Intranet Application servers 60, and the data warehouses, legacy systems, etc. 80.

A customer workstation 51 employs a Web Browser 50 implementing client applications responsible for presentation and front-end services. Its functions include providing a user interface to various MCI services and supporting communications with MCI's Intranet web server cluster 44. As illustrated in FIG. 3, and more specifically described in the above-mentioned, co-pending U.S. patent application

Ser. No. 09/159,515, filed Sep. 24, 1998 (issued as U.S. Pat. No. 6,115,040 on Sep. 5, 2000) entitled GRAPHICAL USER INTERFACE FOR WEB ENABLED APPLICATIONS, the client tier software is responsible for presentation services to the customer and generally includes a web browser 50 and additional object-oriented programs residing in the client workstation platform 51. The client software is generally organized into a component architecture with each component generally comprising a specific application, providing an area of functionality. The applications generally are integrated using a "backplane" services layer 52 which provides a set of services to the application objects which provide the front end business logic and manages their launch. The networkMCI Interact common set of objects provide a set of services to each of the applications such as: 1) session management; 2) application launch; 3) inter-application communications; 4) window navigation among applications; 5) log management; and 6) version management.

The primary common object services include: graphical user interface (GUI); communications; printing; user identity, authentication, and entitlements; data import and export; logging and statistics; error handling; and messaging services.

FIG. 4 is a diagrammatic example of a backplane architecture scheme illustrating the relationship among the common objects. In this example, the backplane services layer 52 is programmed as a Java applet which can be loaded and launched by the web browser 50. With reference to FIG. 4, a typical user session starts with a web browser 50 creating a backplane 52, after a successful login. The backplane 52, inter alia, presents a user with an interface for networkMCI Interact application management. A typical user display provided by the backplane 52 may show a number of applications the user is entitled to run, each application represented by buttons depicted in FIG. 4 as buttons 58a,b,c selectable by the user. As illustrated in FIG. 4, upon selection of an application, the backplane 52 launches that specific application, for example, Service Inquiry 54a or Alarm Monitor 54b, by creating the application object. In processing its functions, each application in turn, may utilize common object services provided by the backplane 52. FIG. 4 shows graphical user interface objects 56a,b created and used by a respective application 54a,b for its own presentation purposes.

FIG. 5 illustrates an example client GUI presented to the client/customer as a browser web page 71 providing, for example, a suite 70 of network management reporting applications including: MCI Traffic Monitor 72; an alarm monitor 73; a Network Manager 74 and Intelligent Routing 75. Access to network functionality is also provided through Report Requester 76, which provides a variety of detailed reports for the client/customer and a Message Center 77 for providing enhancements and functionality to traditional e-mail communications.

As shown in FIGS. 3 and 4, the browser resident GUI of the present invention implements a single object, COBackPlane which keeps track of all the client applications, and which has capabilities to start, stop, and provide references to any one of the client applications.

The backplane 52 and the client applications use a browser 50 such as the Microsoft Explorer versions 4.0.1 or higher for an access and distribution mechanism. Although the backplane is initiated with a browser 14, the client applications are generally isolated from the browser in that they typically present their user interfaces in a separate frame, rather than sitting inside a Web page.

The backplane architecture is implemented with several primary classes. These classes include COBackPlane, COApp, COAppImpl, COParam, and COAppFrame classes. COBackPlane 52 is an application backplane which launches the applications 54a, 54b, typically implemented as COApp. COBackPlane 52 is generally implemented as a Java applet and is launched by the Web browser 50. This backplane applet is responsible for launching and closing the COApps.

When the backplane is implemented as an applet, it overrides standard Applet methods init(), start(), stop() and run(). In the init() method, the backplane applet obtains a COUser user context object. The COUser object holds information such as user profile, applications and their entitlements. The user's configuration and application entitlements provided in the COUser context are used to construct the application toolbar and Inbox applications. When an application toolbar icon is clicked, a particular COApp is launched by launchApp() method. The launched application then may use the backplane for inter-application communications, including retrieving Inbox data.

The COBackPlane 52 includes methods for providing a reference to a particular COApp, for interoperation. For example, the COBackPlane class provides a getApp() method which returns references to application objects by name. Once retrieved in this manner, the application object's public interface may be used directly.

The use of a set of common objects for implementing the various functions provided by the system of the present invention, and particularly the use of browser based objects to launch applications and pass data therebetween is more fully described in the above-referenced, co-pending application GRAPHICAL USER INTERFACE FOR WEB ENABLED APPLICATIONS.

As shown in FIG. 3, the aforesaid objects will communicate the data by establishing a secure TCP messaging session with one of the DMZ networkMCI Interact Web servers 44 via an Internet secure communications path 32 established, preferably, with a secure sockets SSL version of HTTPS. The DMZ networkMCI Interact Web servers 44 function to decrypt the client message, preferably via the SSL implementation, and unwrap the session key and verify the users session. After establishing that the request has come from a valid user and mapping the request to its associated session, the DMZ Web servers 44 will re-encrypt the request using symmetric encryption and forward it over a second socket connection 33 to the dispatch server 46 inside the enterprise Intranet.

As described in greater detail in commonly owned, co-pending U.S. patent application Ser. No. 09/159,514, filed Sep. 24, 1998 entitled SECURE CUSTOMER INTERFACE FOR WEB-BASED DATA MANAGEMENT, the contents and disclosure of which is incorporated by reference as if fully set forth herein, a networkMCI Interact session is designated by a login, successful authentication, followed by use of server resources, and logout. However, the world-wide web communications protocol uses HTTP, a stateless protocol, each HTTP request and reply is a separate TCP/IP connection, completely independent of all previous or future connections between the same server and client. The nMCI Interact system is implemented with a secure version of HTTP such as S-HTTP or HTTPS, and preferably utilizes the SSL implementation of HTTPS. The preferred embodiment uses SSL which provides a cipher speech message which provides server authentication during a session. The preferred embodiment further associates a given HTTPS request with a logical session which is initiated and tracked

by a "cookie jar server" 48 to generate a "cookie" which is a unique server-generated key that is sent to the client along with each reply to a HTTPS request. The client holds the cookie and returns it to the server as part of each subsequent HTTPS request. As desired, either the Web servers 44, the cookie jar server 48 or the Dispatch Server 46, may maintain the "cookie jar" to map these keys to the associated session. A separate cookie jar server 48, as illustrated in FIG. 3 has been found desirable to minimize the load on the dispatch server 46. This form of session management also functions as an authentication of each HTTPS request, adding an additional level of security to the overall process.

As illustrated in FIG. 3, after one of the DMZ Web servers 44 decrypts and verifies the user session, it forwards the message through a firewall 55b over a TCP/IP connection 33 to the dispatch server 46 on a new TCP socket while the original socket 32 from the browser is blocking, waiting for a response. The dispatch server 46 will unwrap an outer protocol layer of the message from the DMZ services cluster 44, and will reencrypt the message with symmetric encryption and forward the message to an appropriate application proxy via a third TCP/IP socket 37. While waiting for the proxy response, all three of the sockets 32, 33, 37 will be blocking on a receive. Specifically, once the message is decrypted, the wrappers are examined to reveal the user and the target middle-tier (Intranet application) service for the request. A first-level validation is performed, making sure that the user is entitled to communicate with the desired service. The user's entitlements in this regard are fetched by the dispatch server 46 from StarOE server 69 at login time and cached.

If the requester is authorized to communicate with the target service, the message is forwarded to the desired service's proxy. Each application proxy is an application specific daemon which resides on a specific Intranet server, shown in FIG. 3 as a suite of mid-range servers 60. Each Intranet application server of suite 60 is generally responsible for providing a specific back-end service requested by the client, and, is additionally capable of requesting services from other Intranet application servers by communicating to the specific proxy associated with that other application server. Thus, an application server not only can offer its browser a client to server interface through the proxy, but also may offer all its services from its proxy to other application servers. In effect, the application servers requesting service are acting as clients to the application servers providing the service. Such mechanism increases the security of the overall system as well as reducing the number of interfaces.

The network architecture of FIG. 3 may also include a variety of application specific proxies having associated Intranet application servers including: a StarOE proxy for the StarOE application server 69 for handling authentication order entry/billing; an Inbox proxy for the Inbox application server 61, which functions as a container for completed reports, call detail data and marketing news messages; a Report Manager Proxy capable of communicating with a system-specific Report Manager server 62 for generating, managing and scheduling the transmission of customized reports including, for example: call usage analysis information provided from the StarODS server 63; network traffic analysis/monitor information provided from the Traffic view server 64; virtual data network alarms and performance reports provided by Broadband server 65; trouble tickets for switching, transmission and traffic faults provided by Service Inquiry server 66; and toll free routing information provided by Toll Free Network Manager server 67.

As partially shown in FIG. 3, it is understood that each Intranet server of suite 60 communicates with one or several consolidated network databases which include each customer's network management information and data. In the present invention the Services Inquiry server 36 includes communication with MCI's Customer Service Management legacy platform 80(a). Such network management and customer network data is additionally accessible by authorized MCI management personnel. As shown in FIG. 3, other legacy platforms 80(b), 80(c) and 80(d) may also communicate individually with the Intranet servers for servicing specific transactions initiated at the client browser. The illustrated legacy platforms 80(a)-(d) are illustrative only and it is understood other legacy platforms may be interpreted into the network architecture illustrated in FIG. 3 through an intermediate midrange server 60.

Each of the individual proxies may be maintained on the dispatch server 46, the related application server, or a separate proxy server situated between the dispatch server 46 and the midrange server 30. The relevant proxy waits for requests from an application client running on the customer's workstation 50 and then services the request, either by handling them internally or forwarding them to its associated Intranet application server 60. The proxies additionally receive appropriate responses back from an Intranet application server 60. Any data returned from the Intranet application server 60 is translated back to client format, and returned over the Internet to the client workstation 50 via the Dispatch Server 46 and at one of the web servers in the DMZ Services cluster 44 and a secure sockets connection. When the resultant response header and trailing application specific data are sent back to the client browser from the proxy, the messages will cascade all the way back to the browser 14 in real time, limited only by the transmission latency speed of the network.

The network MCI Interact middle tier software includes a communications component offering three (3) types of data transport mechanisms: 1) Synchronous; 2) Asynchronous; and 3) Bulk transfer. Synchronous transaction is used for situations in which data will be returned by the application server 60 quickly. Thus, a single TCP connection will be made and kept open until the full response has been retrieved.

Asynchronous transaction is supported generally for situations in which there may be a long delay in application server 60 response. Specifically, a proxy will accept a request from a customer or client 50 via an SSL connection and then respond to the client 50 with a unique identifier and close the socket connection. The client 50 may then poll repeatedly on a periodic basis until the response is ready. Each poll will occur on a new socket connection to the proxy, and the proxy will either respond with the resultant data or, respond that the request is still in progress. This will reduce the number of resource consuming TCP connections open at any time and permit a user to close their browser or disconnect a modem and return later to check for results.

Bulk transfer is generally intended for large data transfers and are unlimited in size. Bulk transfer permits cancellation during a transfer and allows the programmer to code resumption of a transfer at a later point in time.

FIG. 6 is a diagram depicting the physical network MCI Interact system architecture 100. As shown in FIG. 6, the system is divided into three major architectural divisions including: 1) the customer workstation 50 which include those mechanisms enabling customer connection to the Secure web servers 44; 2) a secure network area 47, known as the DeMilitarized Zone "DMZ" set aside on MCI pre-

11

mises double firewalled between the both the public Internet 85 and the MCI Intranet to prevent potentially hostile customer attacks; and, 3) the MCI Intranet Midrange Servers 60 and Legacy Mainframe Systems 80 which comprise the back end business logic applications.

As illustrated in FIG. 6, the present invention includes a double or complex firewall system that creates a "demilitarized zone" (DMZ) between two firewalls 55a, 55b. In the preferred embodiment, one of the firewalls 55b includes port specific filtering routers, which may only connect with a designated port address. For example, router 84 (firewall 55a) may connect only to the addresses set for the HydraWeb® (or web servers 44) within the DMZ, and router 86 (firewall 55b) may only connect to the port addresses set for the dispatch server 46 within the network. In addition, the dispatch server 46 connects with an authentication server, and through a proxy firewall to the application servers. This ensures that even if a remote user ID and password are hijacked, the only access granted is to one of the web servers 44 or to intermediate data and privileges authorized for that user. Further, the hijacker may not directly connect to any enterprise server in the enterprise Intranet beyond the DMZ, thus ensuring internal company system security and integrity. Even with a stolen password, the hijacker may not connect to other ports, root directories or application servers within the enterprise system, and the only servers that may be sabotaged or controlled by a hacker are the web servers 44.

The DMZ 47 acts as a double firewall for the enterprise Intranet because of the double layer of port specific filtering rules. Further, the web servers 44 located in the DMZ never store or compute actual customer sensitive data. The web servers only transmit the data in a form suitable for display by the customer's web browser. Since the DMZ web servers do not store customer data, there is a much smaller chance of any customer information being jeopardized in case of a security breach. In the preferred embodiment, firewalls or routers 84, 86 are a combination of circuit gateways and filtering gateways or routers using packet filtering rules to grant or deny access from a source address to a destination address. All connections from the internal application servers are proxied and filtered through the dispatcher before reaching the web servers 44. Thus it appears to any remote site, that the connection is really with the DMZ site, and identity of the internal server is doubly obscured. This also prevents and direct connection between any external and any internal network or intranet computer.

The filtering firewalls 55(a), (b) may also pass or block specific types of Internet protocols. For example, FTP can be enabled only for connections to the In-Box server 61, and denied for all other destinations. SMTP can also be enabled to the In-Box server, but Telnet denied. The In-box server 61 is a store and forward server for client designated reports, but even in this server, the data and meta-data are separated to further secure the data, as will be described.

As previously described, the customer access mechanism is a client workstation 51 employing a Web browser 50 for providing the access to the networkMCI Interact system via the public Internet 85. When a subscriber connects to the networkMCI Interact Web site by entering the appropriate URL, a secure TCP/IP communications link 32a is established to one of several Web servers 44 located inside a first firewall 55a in the DMZ 47. Preferably at least two web servers are provided for redundancy and failover capability. In the preferred embodiment of the invention, the system employs SSL encryption so that communications in both directions between the subscriber and the networkMCI Interact system are secure.

12

In the preferred embodiment, all DMZ Secure Web servers 44 are preferably DEC 4100 systems having Unix or NT-based operating systems for running services such as HTTPS, FTP, and Telnet over TCP/IP. The web servers may be interconnected by a fast Ethernet LAN running at 100 Mbit/sec or greater, preferably with the deployment of switches within the Ethernet LANs for improved bandwidth utilization. One such switching unit included as part of the network architecture is a HydraWEB® unit 82, manufactured by HydraWEB Technologies, Inc., which provides the DMZ with a virtual IP address so that subscriber HTTPS requests received over the Internet will always be received. The HydraWeb® unit 82 implements a load balancing algorithm enabling intelligent packet routing and providing optimal reliability and performance by guaranteeing accessibility to the "most available" server. It particularly monitors all aspects of web server health from CPU usage, to memory utilization, to available swap space so that Internet/ Intranet networks can increase their hit rate and reduce Web server management costs. In this manner, resource utilization is maximized and bandwidth (throughput) is improved. It should be understood that a redundant HydraWeb® unit may be implemented in a Hot/Standby configuration with heartbeat messaging between the two units (not shown). Moreover, the networkMCI Interact system architecture affords web server scaling, both in vertical and horizontal directions. Additionally, the architecture is such that new secure web servers 44 may be easily added as customer requirements and usage increases.

As shown in FIG. 6, the most available Web server 44 receives subscriber HTTP requests, for example, from the HydraWEB® 82 over a connection 44b and generates the appropriate encrypted messages for routing the request to the appropriate MCI Intranet midrange web server over connection 44a, router 86 and connection 44b. Via the HydraWeb® unit 82, a TCP/IP connection 38 links the Secure Web server 44 with the MCI Intranet Dispatcher server 46.

Further as shown in the DMZ 47 is a second RTM server 92 having its own connection to the public Internet via a TCP/IP connection 88. As described in co-pending U.S. patent application Ser. No. 09/159,516, filed Sep. 24, 1998 entitled INTEGRATED PROXY INTERFACE FOR WEB BASED TELECOMMUNICATIONS MANAGEMENT TOOLS, this RTM server provides real-time session management for subscribers of the networkMCI Interact Real Time Monitoring system. An additional TCP/IP connection 88a links the RTM Web server 92 with the MCI Intranet Dispatcher server 46. As further shown in FIG. 6, a third router 87 is provided for routing encrypted subscriber messages from the RTM Web server 92 to the Dispatcher server 46 inside the second firewall. Although not shown, each of the routers 86, 87 may additionally route signals through a series of other routers before eventually being routed to the nMCI Interact Dispatcher server 46. In operation, each of the Secure servers 44 function to decrypt the client message, preferably via the SSL implementation, and unwrap the session key and verify the users session from the COUser object authenticated at Logon.

After establishing that the request has come from a valid user and mapping the request to its associated session, the Secure Web servers 44 will re-encrypt the request using symmetric RSA encryption and forward it over a second socket connection 38 to the dispatch server 46 inside the enterprise Intranet.

As described herein, and in greater detail in co-pending U.S. patent application Ser. No. 09/159,695, filed Sep. 24,

1998 the data architecture component of the network MCI Interact system is focused on the presentation of real time (un-priced) call detail data, such as provided by MCI's TrafficView Server 64, and priced call detail data and reports, such as provided by MCI's StarODS Server 63 in a variety of user selected formats.

All reporting is provided through a Report Requestor GUI application interface which support spreadsheet, a variety of graph and chart types, or both simultaneously. For example, the spreadsheet presentation allows for sorting by any arbitrary set of columns. The report viewer may also be launched from the inbox when a report is selected.

A common database may be maintained to hold the common configuration data which can be used by the GUI applications and by the mid-range servers. Such common data will include but not be limited to: customer security profiles, billing hierarchies for each customer, general reference data (states, NPA's, Country codes), and customer specific pick lists: e.g., ANI's, calling cards, etc. An MCI Internet StarOE server will manage the data base for the common configuration of data.

Report management related data is also generated which includes 1) report profiles defining the types of reports that are available, fields for the reports, default sort options and customizations allowed; and 2) report requests defining customer specific report requests including report type, report name, scheduling criteria, and subtotal fields. This type of data will be resident in an Inbox server database and managed by the Inbox server.

The Infrastructure component of the nMCI Reporting system includes means for providing secure communications regardless of the data content being communicated. As described in detail in above-referenced, co-pending U.S. patent application Ser. No. 09/159,514, filed Sep. 24, 1998 the nMCI Interact system security infrastructure includes: 1) authentication, including the use of passwords and digital certificates; 2) public key encryption, such as employed by a secure sockets layer (SSL) encryption protocol; 3) firewalls, such as described above with reference to the network architecture component; and 4) non-repudiation techniques to guarantee that a message originating from a source is the actual identified sender. One technique employed to combat repudiation includes use of an audit trail with electronically signed one-way message digests included with each transaction.

Another component of the nMCI Interact infrastructure includes order entry, which is supported by the Order Entry ("StarOE") server. The general categories of features to be ordered include: 1) Priced Reporting; 2) Real-time Reporting; 3) Priced Call Detail; 4) Real Time Call Detail; 5) Broadband SNMP Alarming; 6) Broadband Reports; 7) Inbound RTM; 8) Outbound RTM; 9) Toll Free Network Manager; and 10) Call Manager. The order entry functionality is extended to additionally support 11) Event Monitor; 12) Service Inquiry; 13) Outbound Network Manager; 14) Portfolio; and, 15) Client View.

The Self-monitoring infrastructure component for nMCI Interact is the employment of mid-range servers that support SNMP alerts at the hardware level. In addition, all software processes must generate alerts based on process health, connectivity, and availability of resources (e.g., disk usage, CPU utilization, database availability).

The Metrics infrastructure component for nMCI Interact is the employment of means to monitor throughput and volumes at the Web servers, dispatcher server, application proxies and mid-range servers. Metrics monitoring helps in the determination of hardware and network growth.

To provide the areas of functionality described above, the client tier 50 is organized into a component architecture, with each component providing one of the areas of functionality. As explained in further detail in co-pending U.S. patent application Ser. No. 09/239,115, the client-tier software is organized into a "component" architecture supporting such applications as inbox fetch and inbox management, report viewer and report requester, TFNM, Event Monitor, Broadband, Real-Time Monitor, and system administration applications. Further functionality integrated into the software architecture includes applications such as Outbound Network Manager, Call Manager, Service Inquiry and Client View.

The present invention relates to a data warehousing infrastructure for the StarODS Priced call detail data reporting component of the nMCI Interact system 200. The diagram of FIG. 7 illustrates, at a very high level, the systems involved in modifying and delivering BDRs to the StarODS priced call detail data reporting system, and in requesting, creating and delivering reports to the customer based upon those records. The systems and their functions are described in greater detail below.

As shown in FIG. 7, the StarODS data management architecture, which is integrated with the StarWRS component of the nMCI Interact architecture, comprises a data warehousing infrastructure 400 for providing customers with their priced call detail data. Although the description herein pertains to priced data, it should be understood that the principles described herein could apply to any type of data, e.g., traffic call detail data.

Preferably, the StarODS system provides reporting and customization of priced call detail data and implements a data marts approach for maintaining the data used for customer reporting. StarODS stores and incrementally processes customer's priced data contained in call detail records, and loads this processed data in operational data stores or data marts. From these data marts customer's priced reporting data are provided to customers on a daily basis via the StarWRS reporting system in the manner as described in co-pending U.S. patent application Ser. No. 09/159,684, filed Sep. 24, 1998 entitled INTEGRATED PROXY INTERFACE FOR WEB-BASED DATA MANAGEMENT TOOL, the contents and disclosure of which is incorporated by reference as if fully set forth herein.

As shown in FIG. 7, the data warehousing infrastructure 400 comprises the following system components: a traffic component 405 for receiving call detail records, sorting CDRs into billable records, error and suspense records, expands records, CustID's, prices at tariff; 2) a National Commercial Billing System "NCBS" mainframe process 410 that performs pricing at tariff for nMCI Interact virtual network ("Vnet") and Vision customers and, processes by runstream at one or more datacenters; 3) a Tollfree Billing mainframe process 420 that performs pricing at tariff for Tollfree customers and, processes by runstream at one or more datacenters; 4) Common Data Gateway (CDG) 430 comprising: a) an Extract process 500 for creating selection tables including all current nMCI Interact customers, compressing files for transmission to service centers, and extracting (Priced Reporting enabled) records from divisions or runstreams; and, b) a Harvesting component 600 including processes for creating dimension tables based on data within selected BDRs, applying business rules to the data, transforming the data into centralized fact table, creating load files for data marts, and compressing files for transmission; 5) Operational Data Store (ODS) component 450, including a process 465 for loading transformed billing detail records

as a centralized fact table in one or more data marts storage devices, and integrating both static and dynamic dimension tables 460 according to a star-schema structure so that on-demand reports may be efficiently produced; 6) data marts 470 for storing the billing detail records in a fact table database, e.g., Informatica, organized in a star-schema structure to facilitate priced call detail reporting; 7) a Decision Support Server 475 executing a combination of logic programs such as C4+ and Information Advantage® software for use as the reporting engine. This component reads metadata, translates into queries, runs queries against harvested data fact tables in data marts, formats query results into a format readable by Message Center viewers, transmits complete reports to directory on Inbox server, and, additionally, performs cost estimation, scheduling, transaction logging and generates report metrics; and, 8) Talarian Smart Sockets interface between the decision support server and the StarWRS report requester reporting system comprising messaging middleware used to coordinate report requests transmitted from StarWRS to DSS.

Additionally, as shown in FIG. 7, other external systems and applications may interface with the common data gateway component 430 including: Cyclone Billing system 422a and Concert Virtual Network Services 422b which provide additional billing detail records; and, a calling area database 425 which provides geographical reference information, i.e., identify city, state and country information.

Additionally shown in FIG. 7, and explained in greater detail in co-pending U.S. patent application Ser. No. 09/159,684, filed Sep. 24, 1998, is the StarWRS web based reporting system 200 including: Report Manager for storing report definitions, metadata; Report requester for providing the interface used by the customer to enter report criteria, and submit report metadata to Decision Support Server; and, StarWRS Inbox or Message Center for holding completed reports, providing an interface notifying customer that reports are available, and supplying metadata to a Report Viewer component so that user can view a report.

As will be explained, the data warehousing process is a two fold approach including the provision of incremental, daily updates to the data marts for customer's priced billing data, and, a monthly audit/reconciliation to ensure that daily totals for priced data closely tracks monthly totals, within a predetermined variance, for the same data that may have been subject to re-pricing efforts. For example, daily priced call detail data totals, e.g., total call amount, duration and count, for a customer may match to within 5 percent variance, of their corresponding monthly data that is used for invoicing.

FIG. 8 illustrates a detailed overview of the extracting process 500 of the StarODS CDG component 430 of the data warehousing infrastructure of the invention. With regard to FIGS. 8 and 9, the call processing/traffic component 405 performs the following steps: receiving binary switch records from switch Adjunct Processors and Storage and Configuration elements ("SAVE") 402; translating switch records from binary to EBCDIC format so they can be read, for example, by an IBM mainframe; determining the date and time of each switch record; expanding on the record; calculating call duration and adds to each record; validating each record to determine if it is billable, and dropping unbillable records; validating each record to determine if it is billable, and dropping unbillable records; determining the switch type and adding NPA/NXX information; determining the call type; determining which division each record belongs to, i.e., which data center from among one or more

processing datacenters should receive the record if a plurality of data centers exist; determining service type, performing distance calculations; identifying and segregating any billable calls with errors and adding these records to a "error" file, for example, for future processing; adding the account number and billing cycle to each record; and, pricing all records destined for NCBS 410 at tariff.

The NCBS component particularly produces new daily call detail feed files 522a, 522b for MCI Vnet/Vision customers, e.g., generated by product and divisional runstream. Tollfree processing system 420 also creates new daily call detail feed files 522c by product and divisional runstream.

With regard to the daily mainframe server extract process 500 shown in FIG. 8, the first step 502 is triggered by the availability of a customer selection list file which, in the preferred embodiment, represents those nMCI Interact customers entitled to receive priced call detail data via their web browsers and consequently, which customer BDRs are to be extracted. Specifically, the CDG component is a mainframe process running at a primary datacenter which first executes an FTP pull of the customer list file from the nMCI Interact StarOE server 280, as indicated at step 504, and receives the updated customer file list at step 505. This customer list file may be used to drive subsequent extract processing at other data centers (not shown). In the preferred embodiment, the extract granularity in the daily customer list file is to the service location level, allowing extract of subsets of customer data down to service location level. Once retrieved from the StarOE server 280, the customer selection list may be transferred via network data mover "NDM" to the CDG extract processes which may be awaiting this file at other datacenters to trigger BDR extract processes at those other centers, assuming daily billing feeds are available at those data centers. In the preferred embodiment, the StarOE customer selection list file is used to extract Vnet, Vision, and Tollfree daily billing call data records for nMCI Interact/StarODS priced reporting customers from the daily feeds from Tollfree/NCBS. Daily BDR's are available by division for Tollfree and by runstream from NCBS (Vision/VNET).

The various divisional/runstream files may be consolidated in each CDG extract process, to generate a single extracted daily data file 523 per division in a data center. Furthermore, as indicated at step 506, a customer delta file is created which comprises a list of the new nMCI Interact priced reporting customers who were not in the previous day's customer selection list file. This customer delta file is backed up to step 508 and, as indicated at step 510, may be transmitted to other data centers, e.g., for reporting purposes, or, to enable synchronization among the one or more datacenters.

As shown in FIG. 8, a determination is made at step 511 as to whether the number of new customers in the customer delta file has exceeded a predetermined threshold, e.g., greater than a 20% addition. If a threshold is exceeded, the extract process terminates until a manual override is performed to check out confirm the variance. Otherwise, the process continues at step 512.

At step 512, the customer file may be further broken down into billing products, e.g., Toll free, Vnet, and having product identifiers such as Corp ID or Service ID, and may be loaded into a virtual storage access ("VSAM") system for enabling quick retrieval, as indicated at step 514.

As CDG mainframe daily and monthly extract processing are triggered, on a division basis, by the completion of the appropriate corresponding billing daily or monthly division job(s), CDG maintains an audit point based on data extracted

on a daily or monthly basis. Audit totals are used to match back to billing summary totals which may trigger a monthly data replace operation, as will hereinafter be explained in greater detail. CDG also provides summary totals by customer to be used to verify data harvesting and database load totals.

Next, as indicated at step 523, the raw billing detail records are input from the runstreams, e.g., Toll free, and a reconciliation is performed at step 528 to ensure that all BDR records from the runstreams are received files for the current customer list. Then, as indicated at step 530, the extract process is performed for the BDR datasets as they become available from the run streams. In the preferred embodiment, a SyncSort process is used to select only the customer data needed by the nMCI Interact StarODS system based on the customer selection list. Particularly, certain fields of the daily feed file are rearranged, and a value, e.g., determined by the production run date, is written to each daily record processed in the billing period and is used by the DSS in the data load process to replace daily records with their audited, final monthly counterparts, as will be explained. As part of extraction, the fields in each record are rearranged to the format usable by the DSS server, and identical to those produced by the daily process. In the event that invalid data types are encountered, the invalid data is replaced with designated valid values and invalid fields of any records may be blocked out based upon pre-determined criteria. Additionally input to the BDR record is an invoicing period value, e.g., "bill_period" which may be used as a text tag, to facilitate the above-mentioned monthly replace operation. More particularly, BDR records are rearranged so that the Corp ID, Service ID, Invoicing Period, Call Minutes and Call Amount are the first five fields of each record. Output BDR records are adjusted to have a structure and size which corresponds to the monthly data and, preferably are of constant length. In the preferred embodiment, the output BDR records have a constant length per record, with the output file defined as a variable blocked file with a record length of, for example, 1024 bytes. This allows dissimilar BDR output files to be concatenated for later reporting purposes, i.e., records produced by Vision may be concatenated and reported with records from VNET or Tollfree.

As previously mentioned, for the monthly audit process, the final step is the production of an audit file which is used as a control by the data Harvest process. This is accomplished by reading in the processed BDR extract file and producing an audit file comprising all of the Corp ID & Service ID combinations for which BDR data was extracted, and the record counts, total minutes and total amounts for these records.

As further shown in the FIG. 8, there is shown a CADB/Geography build process 518 which receives a calling area database file (CADB) file 426 input from the StarOE server and a Vnet billing system City Name file input to determine if new records need to be added to a NPA/NXX dimension table. This is accomplished by first SyncSorting the CADB file down to unique country code, NPA, Nxx, bill code, City Name, and State Code combinations. The CADB file is read and compared to the NPA/NXX table and the country code (CC), NPA, NXX, City Name, and State Name are used to see if that exact combination exists. If it does not, then it will be added to the NPA/NXX table and added to the dimension add file. A Key Sequence file (not shown) may be used to generate the key. Then, the billing system name file is read and bill code, City Name and State Code, will be compared against the NPA/NXX tables bill code, City2, and State Code. If the exact combination does not exist then it

will be added to the NPA/NXX table and the dimension add file, using the Key Sequence file to generate the key. Continuing with the process, a SyncSort may be used to sort the NPA/NXX table first by NPA/Nxx, then by City2. Finally, the dimension add file is FTP'd to the database server.

As mentioned, the ODS Data Harvesting Agent 600, e.g., Prism™ Warehouse Executive, is the component of the Priced Reporting system that focuses on the transformations and manipulations of billing data. Harvesting is the set of functionality that describes the form raw billing data must assume to be useable by customers to satisfy their questions and analysis requests. Common data gateway (CDG) mainframe system extracts raw Billing Detail Records (BDRs) for those entitled NMCI Interact/ODS customers, and Data Harvesting transforms the Billing Detail Records based on a set of business rules applied to the call data. The final result of harvesting is call detail data ready to be loaded into the ODS data marts for customer access and analysis.

Like the extract process, daily data harvesting is accomplished on a divisional/runstream basis, distributed across one or more datacenter mainframe sites. It is triggered by the completion of the corresponding division extract in the CDG mainframe extract process at the data center. During data harvesting the extracted billing call detail records are transformed based on the star-schema data model, which incorporates a central Fact table of call records surrounded by appropriately (foreign key) referenced satellite dimension tables. All dimensional table elements are available prior to harvesting as dimension-based lookups and, preferably, harvesting logic enables creation of new dimension keys for any dimension value(s) not found in dimension lookup tables.

As illustrated in FIG. 9, the data model 759 implemented in StarODS is a dimensional or "star schema" model, including a central fact table multiply joined to a number of attendant tables, i.e., dimensions. The relationships between the fact table and the dimensional tables are either enforced through keys, which may be generated, or as lookup codes.

As shown in FIG. 9, the central fact table 761 is known as "Perspective Base" and provides access to a collection of attributes or facts concerning a call. The dimensional tables include the following: an Access Termination table 762 comprising data indicating whether a call was charged to recipient (inbound) or originator (outbound); an Access Type table 764 comprising data indicating the type of access (for outbound calls) or egress (for inbound calls) characteristics of a call; a Billing Corp table 766 comprising data indicating the hierarchical status of a customer for the purposes of billing charges for products and features; a Toll Free Number table 768 comprising toll-free numbers i.e., 800# or 888# (in the USA); a Product Type table 769 comprising data indicating the product for which services are bundled for the purpose of invoicing; a GMT table 771 comprising date and time data adjusted to the Greenwich Mean Time Zone; a LST table 773 comprising date and time data adjusted to the local MCI switch which permitted access to the MCI network; an Orig_Geo table 776 comprising data indicating the geographic characteristics of a call's origination; a Term_Geo table 777 comprising data indicating the geographic characteristics of a call's termination; a Report Geo table 778 comprising data indicating the geographic characteristics of a call's origination or termination; an Idacae table 779 comprising data indicating a customer's defined id and/or accounting code; a Data Stream table 781 comprising data relating to the line speed characteristics of a data (non-voice) call; a Pay Phone table 782

comprising data denoting calls originating from a payphone; a Usage table 783 comprising data indicating the geographic attributes of a call which affect tariff rates; an EVS Product table 784 comprising data representing Enhanced Voice Services products; a Directory Assistance table 786 comprising data indicating those calls requesting Directory assistance; a Range table 787 comprising data indicating distance bands a call may fall into; an NCR table 788 indicating Network Call Redirect calls; a Cell Phone table 789 comprising cellular call characteristics data; a VOS table 791 indicating Voice Operator Services calls; a Conference Call table 792 having data pertaining to characteristics of conference calls; a Cross Corp table 793 comprising data indicating inbound cross corporate routing of calls; a Currency table 794 indicating the unit of currency for call prices; a card table 796 comprising data for billing calls to a location that may not be the one which originated the call; an NCT table 797 comprising data representing Network Call Transfers; an Amount Range table 798 indicating call usage ranges based upon amounts; and, a Duration Range table 799 indicating call usage durations based on amounts. This star schema model is optimized for decision support and the retrieval of large amounts of data. Appendix H provides the data attributes of each of these dimension tables. As known, in the dimensional model, the grain of data stored in the fact table determines what level of data can be drilled down into. It should be understood that the grain of the data stored in the Perspective Base table is at the singular call level.

In the preferred embodiment, all dimension lookup tables must be built. Dimensions, such as Access Type Dimension, Product Type Dimension, and Date-time Dimension, are manually built one time prior to migration to production and are infrequently, if ever, updated, and are hence referred to as static dimensions. The Access Type and Product Type Dimensions are updated or added to only as new business requirements are brought to light. For example, the Access Type Dimension is updated only if new access types are created or added and, the Product Type Dimension is updated only if new products are supported by nMCI Interact priced reporting. The Date-Time Dimension on the other hand may be updated, or rebuilt each year.

The remaining dimensions, such as, the Billing Dimension, Geography Dimension, 800 Number Dimension, the Calling Card Dimension, and the ID and accounting codes ("IDACC") Dimension (if implemented), are dynamically built and maintained by the harvesting programs. Thus, if a lookup to one of these dynamic dimensions encounters a "Not Found" condition, the offending call record may be pending to an unprocessed calls file in the same format as input by the harvesting program. In that scenario, the dimensional value that was not available in the lookup dimension is written to a dimension delta file which may be later used to update that dynamic dimension. The pending and unprocessed call records may then be rerun through the same Harvesting process. In this way, these dynamic dimensions are automatically maintained and updated by harvesting processes. In the preferred embodiment, a new key is generated and the dimension table automatically updated when a dimension lookup can not be found, thus obviating the need to create delta files and running these unprocessed records a second time through harvesting. Note that if a lookup to a static dimension is not found, that record may be rejected.

For example, after availability of the StarOE customer list file, a customer delta file provided during extraction and comprising all new customer list records added since the

previous day's StarOE customer list file, provides the basis for a Billing/Customer Dimension "build" process. That is, since Billing/Customer is a dynamic dimension which incorporates new customer list records into the previous day's Billing/Customer Dimension Lookup File. The newly updated Billing/Customer Dimension Lookup File is subsequently used by the harvesting programs to associate the appropriate Billing/Customer Dimension keys with the corresponding central fact table records being generated.

The association of dimension tables to a centralized fact table through a foreign key structure is created for each defined dimension or dimensional lookup, and is graphically represented in FIG. 10 by the simplified example as follows:

As shown in FIG. 10, each record 602a,b,... of the extracted BDR 604 includes the Corp ID field 605 along with other arranged data pertaining to that particular call, e.g., date, time, call amount. An example billing/customer dimension table 610 and Date dimension table 612 are also shown in FIG. 10. In the example, the billing/customer dimension table include keys 614a, and 614b, corresponding to respective customers Corp ID 1, Corp ID 2. The date dimension table includes date keys 616a and 616b, corresponding to respective dates Jul. 1, 1998 and Jul. 2, 1998. After running the BDR 604 against the dimension tables 610, 612 as shown in FIG. 10, the harvesting process generates a FACT table record 620 having a mix of dimension keys, e.g., 614a, 616a, and data corresponding to the extracted BDR. The combination of these entities represents how data is to be stored in data marts for the operational data store component of StarODS and, further enables subsequent queries against the data to be formed by the decision support servers. In the preferred embodiment, the granularity of the data in the fact table or "Perspective base," is at the singular call level.

According to the invention, harvesting of daily and monthly customer billing data for Priced Reporting may occur in a distributed manner, i.e., across many production data centers or wherever an upstream billing systems and data processing takes place. Like upstream billing systems and the Common Data Gateway, extract processing and data harvesting may be distributed across these data centers by geographic-based divisions and runstreams. Product and frequency typically distinguish harvesting processing. While the basic process logic is similar for all products regardless of whether it involves daily or monthly data harvesting, different dimensional processing as well as distinct business rules implementation are associated with each distinct product. As such, distinct data harvesting program modules are implemented for each of the following: Daily data; Monthly Vnet data; Monthly Vision data; Monthly domestic Tollfree data; Monthly international Tollfree data; and, Monthly CVNS data.

The daily harvesting process will now be described with reference had to FIG. 11. For purposes of explanation, it is assumed that Fact and Dimension tables are being created for Tollfree and/or Vnet/Vision BDR records (daily and monthly). As mentioned, the daily data harvesting process is triggered to run the moment the customer list file from the Common Data Gateway mainframe becomes available. The program is triggered to run the moment the BDR file from the Extract process becomes available. In a first step, the file is opened and the extracted and modified billing detail record is read. Then, the following steps are performed to write dimension data to the (FACT) record, and/or to add a record to each Dimension table: first, as indicated at step 625, a search is made in a "GEO Dimension Table—Originating" where the Geo (originating) Dimension is read.

If a matching record is found, the GEO RECORD KEY is moved to the created fact table (ORIG GEO RECORD KEY); if no record is found, the CDG component preferably generates a new GEO RECORD KEY and moves it to the fact table. Otherwise, spaces may be written to the fact table (ORIG GEO RECORD KEY) and a "DELTA" record is written using GEO KEY. Next, as indicated at step 630, a search is made in a "GEO Dim Table—Terminating" where the Geo Dimension (Terminating) is read. If a matching record is found, the GEO RECORD KEY is moved to the fact table. (TERM GEO RECORD KEY); if no record is found, the CDG component preferably generates a new GEO RECORD KEY and moves it to the fact table. Otherwise, spaces may be written to the fact table (TERM GEO RECORD KEY) and a DELTA record is written using GEO KEY. Next, as indicated at step 635, a search is made in the BILLING Dimension Table where the BILLING Dimension is read. If a matching record is found, the BILLING DIMENSION RECORD KEY is moved to the fact table (BILLING KEY). If no record is found, the CDG harvest component preferably generates a new BILLING RECORD KEY and moves it to the fact table, otherwise spaces may be written to the fact table (BILLING RECORD KEY) and a DELTA record written for the BILLING KEY. Next, as indicated at step 640, a search is made in a Calling Card Dimension Table where the CARD Dimension is read. If a matching record is found, the CARD RECORD KEY is moved to the fact table (CARD KEY). If no record is found, the CDG harvest component preferably generates a new CARD RECORD KEY and moves it to the fact table, otherwise spaces are written to the fact table (CARD RECORD KEY) and a DELTA record is written using CARD KEY. The look-up process continues for each call detail product that is present in the extracted BDR including: a look-up 645 in the local DATETIME Dimension Table (DATETIME KEY); a look-up 650 in the TOLLFREE NUMBER Dimension table (TFNUMBER RECORD KEY); a look-up in the IDAC Dimension table (IDAC RECORD KEY); a look-up in the USAGE Dimension table (USAGE RECORD KEY); a look-up 655 in the GMT Dimension table (GMT RECORD KEY); a look-up 660 in an ACCESS Dimension table (ACCESS RECORD KEY); a look-up 665 in the PRODUCT Dimension table (PRODUCT KEY); a RANGE Dimension table look-up (RANGE RECORD KEY); and, a DURATION Dimension table look-up (DURATION RECORD KEY). With respect to the GMT Dimension table, the "switch_trnk" field and the "switch_trnk_grp" field is taken from the BDR to find an offset, which will be added to, or subtracted from, the time of the call. With respect to the RANGE Dimension table look-up, if a RANGE Dimension Finding Record Value is greater than (">") a RANGE KEY value, then the RANGE RECORD KEY is moved to the fact table. Likewise, if a DURATION Dimension Finding Record Value is greater than (">") a DURATION KEY value, then the DURATION RECORD KEY is moved to the fact table.

As all Dimension tables are populated with keys, the record is written to the FACT Table (FIG. 10). Otherwise, the record is placed into suspense (DELTA) for reprocessing. The process outlined above continues for each BDR record received in the bill stream. If dynamic dimension tables have not been updated, then after all records are read, all dynamic dimension tables are updated with the written deltas. If the file is complete the error file is read, and each record reprocessed to add dimension keys for each record. When the error file has been processed, the harvest process for that file ends and notification is sent to the StarODS database server that "files are ready."

It should be understood that similar daily harvesting procedures are implemented for those BDR records for other products besides Toll free, e.g., Vnet/Vision customers. For VISION/VNET (Monthly or Daily) the following steps are executed for each BDR record to create the Fact and Dimension tables: First, the file is opened and a billing detail record is read: As an example, the following steps are performed to write a billing dimension key to the (FACT) record, or to add a record to each dynamic dimension table: first the following business rules are applied to the BILLING Dimension Table: 1) Moving STK-ENT-ID-CD to BILLING KEY CORP; 2) moving STK-BILL-ID to BILLING KEY BILL (If STK-BILL-ID=SPACES MOVE ZERO); and, 3) moving SVC-LOCN-ID to BILLING KEY SVCLOC. Then, the BILLING Dimension is read. If a matching record is found, the BILLING RECORD KEY is moved to the Fact table (BILLING RECORD KEY). If no record is found, a new key may be generated for input to the fact table and billing dimension table (BILLING RECORD KEY). Each available file from the upstream billing systems is processed as it is available. As a result, multiple runstream files are processed simultaneously, and Decision Support Server 475 (FIG. 6) ftp pulls hundreds of daily files from the harvesting (Common Data Gateway mainframe), as indicated at steps 669a, 669b in FIG. 11.

Following completion of daily data harvesting processing for a given division or runstream but prior to file transfer and load of division harvested data into the priced reporting database(s), i.e., data marts,—all dimension adds for that division or runstream are transferred and loaded into the appropriate database, as indicated at step 666. Other data transformations may occur prior to loading, for instance, conversion of data from IBM Mainframe EBCDIC character set data to mid-range server readable ASCII file format is performed for the output fact table records as indicated at step 665. Additionally, the central BDR fact table records are properly formatted for querying by DSS. Finally, the Data Harvesting Process provides notification to the Priced Reporting database server(s) regarding availability and size of harvested data files. It should be understood that, prior to data loading, an approval is required to ensure that appropriate space is allocated to the databases.

The FTP loading of various daily divisional/runstream harvested data files is initiated as soon as the data harvesting for a division or runstream is completed, as indicated in FIG. 11 at steps 669a,b. Daily harvested data files are transferred or available in the datacenter in the appropriate ASCII file format. Furthermore, loading of a given monthly divisional/runstream harvested data file is triggered by the corresponding formal invoicing approval for that division or runstream. Appendix I is a listing in DDL representing the star topology organization of the fact tables, dimension tables, and their associations in the operational data storage devices or data marts, i.e., an Informix™ database. Included in the file listing of Appendix I, is the organization of the fact table "informix.lopers_base", associations with the above-described dimension tables in a star-schema relation, and, the format for the corresponding customer specific data information, e.g., dialed number, duration, call amount, etc., included in the fact table from the BDRs. The database is provided with ancillary tables used providing additional quality assurance functionality including: a file tracking table "informix.file_tracking" which tracks and maintains all files that are input from harvesting mainframe process; and, request tracking tables "informix.request," "informix.request_hist" and "informix.request_status" which are used to track customer report requests and the status of reports during the report generation process.

With regard to the availability of monthly billing call detail records, the elements of the monthly extract, harvest, load and report production processes are the same as the corresponding daily routines. On days when a monthly processing is occurring, daily processing simultaneously occurs. The major difference is that monthly processing is triggered by the receipt of a monthly BDR file from the appropriate billing system, not the receipt of daily customer list information. At a high level, the sequence is as follows: the monthly extract process runs; the monthly harvest process runs; monthly file sizing is completed; output files are retained pending audit approval of billing system feeds from which call records were extracted and harvested. Information about file names, sizes and locations is retained by the Harvest process 600 for later transmission to StarODS DSS server. After audit approval is received, the data harvesting agent executes notification of ODS. Notification includes availability, name, location and file size. StarODS load process executes file pull for each file and DSS confirms that each compressed file fragment is complete. DSS loads files in the appropriate operational data store or data mart by executing a replace operation of all daily records for that product, invoice period and day with their equivalent records from the monthly file. Granularity of the replace operation is on a daily level, i.e., daily blocks of BDR's are replaced, not individual BDR's. Completion of entire monthly replace operation for specific invoice period and product triggers the DSS server to begin running monthly reports which have been queued up.

It should be understood, that the monthly replace operation is key to providing accurate data for use by DSS in reporting.

More particularly, on a monthly basis, the priced reporting system extracts monthly billing data from the upstream billing systems, including NCBS and Tollfree. From NCBS, monthly division extracts are produced for Vnet and Vision. Monthly division extracts for domestic and international Tollfree are generated from the output of the Tollfree billing cycle. Extract of nMCI Interact specific call detail records from the full output of the billing system is accomplished on a divisional level.

As part of the monthly extract process, a value determined by the production run date is added to the BDR, and is used by the DSS in the data load process to replace daily BDR records with their audited, final monthly counterparts. As part of extraction, fields in each record are re-arranged to the format usable by the DSS, and identical to those produced by the daily process. In the event that invalid data types are encountered, the invalid data is replaced with designated valid values.

In addition, a statistical summary of the results of the Monthly Extract process is produced. This output is compared to the grand total of daily summaries produced by the daily extract process in the same billing period. The sum of daily BDR counts is compared to the grand total of BDR counts for the bill period/product/division, as well as daily counts from the daily process to daily counts per the monthly extract. This is done as a monthly reconciliation to ensure the completeness and accuracy of the monthly file prior to the monthly replace operation which is accomplished by the DSS server 475.

Referring back to FIG. 8, after the daily and monthly harvest and monthly replacement operations have been performed, the data is input to an operational data store component ("ODS") 450 that stores the billing detail records and dimension tables as a data model. This ODS layer 450 is comprised of all data harvested from all applications in the

data harvesting layer 430, and feeds report-supporting data marts 470 in a manner which supports customized data access. The data marts may be engineered to pre-process data, create aggregates, and otherwise perform transformations on the data prior to data mart 465 in order to implement a defined data model, e.g., star schema key structures, fact and dimension tables depicted as block 460. In the preferred embodiment, as shown in FIG. 8, the Operational Data Store 450 includes multiple datamarts 470 each for storing and retrieving daily and monthly priced data on a periodic basis. It primarily is responsible for hosting highly current data, typically at least 72 hours old. In accordance with customer-reporting needs, data marts 470 are partitioned in accordance with partitioning schemes which may be based on customer-ID. Particularly, each data mart is engineered for servicing specific customers or specific product sets, as well as engineered for the specific requirements of the customer/product such as high insert activity, heavy reporting requirements, etc. As data is volatile and changing and may not produce consistent results for the same query launched at multiple times, ODS is engineered for high performance through appropriate storage technologies and parallel processing.

From these data marts customer's priced reporting data can be provided to customers on a daily basis via the StarWRS reporting system as described in detail in above-mentioned co-pending U.S. patent application Ser. No. 09/159,684, filed Sep. 24, 1998. Reporting categories from which a variety of reports can be generated include: a) Financial category—for providing priced data reports relating to longest calls, most expensive calls, Off Peak Calls, payphone report, usage summary, calling card summary, and area code summary for Toll Free, VNET, Vision, and CVNS customers; b) Marketing category—for providing priced data reports relating to country code summary, state summary, frequent numbers, frequent area code summary, frequent state, and frequent cities; c) Telecommunications category—for providing priced data reports relating to call duration summary, IDAC/Supp Code Summary and Call Access Summary for Toll Free, VNET, Vision, CVNS customers; d) Call Center report category—for providing priced data reports relating to most active toll free numbers, Hourly Distribution, Day of Week Distributions, state summary, and country code summary for their Toll Free, VNET, Vision, CVNS customers; e) Monitor Usage—for providing priced data reports relating to longest calls, most expensive calls, most active calling card and most active toll free numbers for their Toll Free, VNET, Vision, CVNS customers; f) Analyze Traffic—area code summary, country code summary, state summary, range summary, city summary, frequent numbers, payphone report, usage summary, calling card summary, IDAC/Supp Code Summary, Day of Week Distributions, Hourly Distribution, Call Access Summary and review calls; and, a g) Check Calling Frequencies category—for reporting on frequent numbers, frequent area code, frequent country codes, frequent state and frequent cities.

Additionally, referring back to FIG. 7 there is provided a decision support server ("DSS") reporting engine component 475 that performs the following functions: 1) receives data access requests from various users in the form of a report request from the StarWRS GUI Report Requestor component; 2) routes the query to the appropriate data marts 470, data warehouse or operational data store; and, 3) responds to the requestor with the result set. The DSS server 475 may also perform cost estimation, agent scheduling, workflow broadcasting interface, and transaction logging functions. In the preferred embodiment, the DSS 475 is a

cluster of DEC (Digital Equipment Corp.) UNIX 8400 servers running Information Advantage® software accessing an Informix database distributed across multiple data marts.

Thus, the present invention is integrated with a client and middle-tier service and application proxy component that enable customers to request, specify, customize, schedule and receive their telecommunications network call detail data and account information in the form of reports that are generated by the various back-end application servers. Referred to herein as "StarWRS," this WWW/Internet Reporting System 200, as shown in FIG. 12, comprises the following components and messaging interfaces:

1) those components associated with the Client GUI front end including a report requestor client application 212, a report viewer client application 215 and, an Inbox client application 210 which implement the logical processes associated with a "Java Client," i.e., employs Java applets launched from the backplane (FIG. 4) that enable the display and creation of reports and graphs based on the fields of the displayed reports, and, allows selection of different reporting criteria and options for a given report; and,

2) those middle-tier server components enabling the above-mentioned reporting functionality including: a Report Manager server 250, a Report scheduler server 260, and an Inbox Server 270. Also shown in FIG. 7 are the system Order Entry client application 280 and a corresponding Order Entry Server 285 supporting the StarWRS reporting functionality as will be described.

Each of these components will now be described with greater particularity hereinbelow.

The Report Manager ("RM") server 250 is an application responsible for the synchronization of report inventory with back-end "Fulfilling" servers 400, 500; retrieval of entitlements, i.e., a user's security profiles, and report pick list information, i.e., data for user report customization options, from the system Order Entry server 280; the transmission of report responses or messages to the Dispatcher server 46 (FIG. 3); the maintenance of the reporting databases; and, the management of metadata used for displaying reports. In the preferred embodiment, the RM server 250 employs a Unix daemon that passively listens for connect requests from the GUI client applications and other back-end servers and deploys the TCP/IP protocol to receive and route requests and their responses. Particularly, Unix stream sockets using the TCP/IP protocol suite are deployed to listen for client connections on a well-known port number on the designated host machine. Customers desiring to submit requests connect to RM 250 via the dispatcher 46 by providing the port number and host name associated with RM 250. For the particular back-end server 400 providing priced reporting data, a Talarian smart socket connection 255 is provided. Request messages received by the RM server are translated into a "metadata" format and are validated by a parser object built into a report manager proxy 250' that services requests that arrive from the GUI front-end. If the errors are found in the metadata input, the RM 250 will return an error message to the requesting client. If the metadata passes the validation tests, the request type will be determined and data will be retrieved in accordance with the metadata request after which a standard response will be sent back to the requesting client. As shown in FIG. 12, interface sockets 252 are shown connecting the Dispatcher server 26 and the RM server 250 and, other socket connections 254, 256 are shown interfacing the RM 250 with respective back end servers 400 and 500. For instance, in

one embodiment, fulfilling server 400 provides a customer's priced billing data through a Talarian smart socket messaging interface 254 to the Report Manager. Additionally, as part of the StarWRS web reporting system 200 shown in FIG. 12, unpriced traffic data may be sent directly to the report manager 250 from the Traffic View server ("TVS") 500, as described in commonly-owned, co-pending U.S. patent application Ser. No. 09/159,684 entitled INTEGRATED PROXY INTERFACE FOR WEB BASED DATA MANAGEMENT REPORTING TOOL. Although not shown in FIG. 12, it should be understood that the RM 250 server may manage reporting data for customer presentation from other back-end and legacy servers including, e.g., Broadband, Toll Free Network Management, and Event Monitor servers, etc. in order to present to a customer these types of billing/management data.

The report manager server additionally utilizes a database 258, such as provided by Informix, to provide accounting of metadata and user report inventory. Preferably, an SQL interface is utilized to access stored procedures used in processing requests and tracking customer reports. A variety of C++ tools and other tools such as Rogue Wave's tools++ are additionally implemented to perform metadata message parsing validation and translation functions.

The Report Manager server 250 additionally includes the scheduling information which is passed to the back-end fulfilling servers 400, 500 and stored by them. At times, the Report Manager will request this information from the fulfilling servers in order to reconcile.

The Report Scheduler ("RS") server component 260 is, in the preferred embodiment, a perpetually running Unix daemon that deploys the TCP/IP protocol to send requests to the back-end fulfilling servers such as the StarODS server 400, or TVS server 500, and receive their responses. More particularly, the RS server 260 is a Unix server program that is designed to handle and process report requests to the fulfilling servers by deploying Unix stream sockets using the TCP/IP protocol suite, and sending the report request to client connections on a well-known port number on the designated host machine. As shown in FIG. 12, interface socket connections 264, 266 are shown interfacing with respective back end servers 400 and 500. In the case of priced billing data from ODS 400, report requests are published by the RS server 260 to a pre-defined subject on the Talarian Server. When handling other incoming messages published by back end servers using Talarian Smart Sockets 4.0, another daemon process is necessary that uses Talarian C++ objects to connect their message queue and extract all messages for a given subject for storage in a database table included in database 263. Each message includes the track number of the report that was requested from the fulfilling server.

From the report scheduler interface, the user may specify the type of reporting, including an indication of the scheduling for the report, e.g., hourly, daily, weekly or monthly. For priced data the user has the option of daily, weekly, or monthly. For real-time, or unpriced data, the user has the option of hourly, daily, weekly or monthly. The report scheduler interface additionally enables a user to specify a page or E-mail account so that an e-mail or page message may be sent to indicate when a requested report is in the Inbox server 270.

As shown in FIG. 12, the report scheduler server 260 interfaces directly with the Report Manager server 250 to coordinate report request processing. It should be understood that the respective report management and scheduling functions could be performed in a single server. An overview

of the report request/scheduling process implemented by StarWRS Report Manager and Report Requestor tools may be found in commonly owned, co-pending U.S. patent application Ser. No. 09/159,409, filed Sep. 24, 1998 entitled INTEGRATED PROXY INTERFACE FOR WEB BASED REPORT REQUESTOR TOOL SET, the contents and disclosure of which is incorporated by reference as if fully set forth herein.

The Inbox Server component 270 serves as the repository where the completed user report data is stored, maintained, and eventually deleted and is the source of data that is uploaded to the client user via the dispatcher over a secure socket connection 272. It is also a Unix program that is designed to handle and process user requests submitted in metadata format using an Informix® database. Once report results are received from the StarODS 400 or any other back-end or fulfilling servers (not shown), the Inbox server 270 requests the metadata from the Report Manager server 250 as indicated by the socket connection 272 in FIG. 12. The metadata is stored in the Inbox server database 273 along with the report results. Thus, if the metadata is required to be changed, it will not interfere with the information needed to display the reports included in the Inbox. Additionally, as shown in FIG. 12, the Inbox server interfaces with the report scheduler to coordinate execution and presentation of reports.

The StarOE server 280 is the repository of user pick lists and user reporting entitlements as shown in database 283. Particularly, it is shown interfacing with the Inbox server 270 and report scheduler servers 260. The Report Manager does not interface with or include metadata for StarOE. It will, however, include information in the report metadata that will tell the Report Requestor it needs to get information (i.e., Pick Lists) from StarOE server 285. Particularly, the StarOE server supports pick lists for the selection of priced data based on the following list: Date, Time (Provide in GMT offset), ID Accounting Code (IDAC)/Supp code, Access Type, Corp ID, Service Location w/Service Location Names, Bill Payer w/Bill Payer Names, 8XX Number, City, State/Province, Numbering Plan Area (NPA), NXX (Exchange code where N=2-9 and X=0-9), and Country Code.

With regard to the front-end client GUI components, the above-mentioned Inbox client application 210 functions as an interface between the client software and the Inbox server 270 for presenting to the customer the various type of reports and messages received at the Inbox including all completed reports, call detail, alarms, and flashes. Preferably, the messages for the user in the inbox is sorted by type (e.g., report, call detail, alarms) and then by report type, report name, date, and time. A more detailed description of the StarWRS Inbox Server component may be found in commonly-owned, co-pending U.S. patent application Ser. No. 09/159,512, filed Sep. 24, 1998 entitled MULTI-THREADED WEB BASED USER IN-BOX FOR REPORT MANAGEMENT, the contents and disclosure of which is incorporated by reference as if fully set forth herein.

Particularly, the Inbox client application uses the services of the backplane (FIG. 4) to launch other applications as needed to process report messages. The inbox will also use the services of the data export objects to provide a save/load feature for inbox messages, and, is used to provide a user-interface for software upgrade/download control. Inbox messages are generated by the versioning services of the backplane; actual downloads will be accomplished by a request through the inbox.

In the preferred embodiment, the inbox client is able to receive information on multiple threads to allow a high

priority message to get through even if a large download is in progress. Typically, the browser is configured to allow more than one network connection simultaneously, i.e., the polling thread on the client uses a separate connection to check for new messages, and start a new thread on a new connection when a new message was detected. In this way, multiple messages may be downloaded simultaneously.

The Report Requestor application 212 is a GUI Applet enabling user interaction for managing reports and particularly includes processes supporting: the creation, deletion, and editing of the user's reports; the retrieval and display of selected reports; the display of selected option data; and the determination of entitlements which is the logical process defining what functionality a user can perform on StarWRS. In the preferred embodiment, a Report request may be executed immediately, periodically, or as "one-shots" to be performed at a later time. As described herein, the report scheduler service maintains a list of requested reports for a given user, and forward actual report requests to the appropriate middle-tier servers at the appropriate time. Additional functionality is provided to enable customers to manage their inventory, e.g., reschedule, change, or cancel (delete) report requests.

The Report Viewer application 215 is a GUI Applet enabling a user to analyze and display the data and reports supplied from the fulfilling servers such as StarODS 400, Traffic View ("TVS") 500, and other systems such as Broadband and toll free network manager. Particularly, the Report Manager 250 includes and provides access to the metadata which is used to tell the Report Requestor what a standard report should look like and the "pick-list" options the user has in order for them to customize the standard report. It is used to tell the Report Viewer client how to display the report, what calculations or translations need to be performed at the time of display, and what further customization options the user has while viewing the report. It additionally includes a common report view by executing a GUI applet that is used for the display and graphing of report data and particularly, is provided with spreadsheet management functionality that defines what operations can be performed on the spreadsheet including the moving of columns, column hiding, column and row single and multiple selection, import and export of spreadsheet data, and printing of spreadsheet, etc. It is also provided with report data management functionality by defining what operations can be performed on the data displayed in a spreadsheet including such dynamic operations as sorting of report data, sub-totalling of report data, etc. Furthermore, the report viewer 215 is provided with functionality enabling the interpretation of metadata; and, functionality enabling communication with the Backplane (FIG. 4). The report viewer application 215 is able to accept messages telling it to display an image or text that may be passed by one of the applications in lieu of report data (e.g., Invoice, Broadband report, etc.).

All reporting is provided through the Report Viewer interface which supports spreadsheet, a variety of graphic and chart types, or both types simultaneously. The spreadsheet presentation allows for sorting by any arbitrary set of columns. The report viewer 215 is launched from the inbox client 210 when a report is selected and may also be launched from the inbox when a report is selected.

By associating each set of report data which is downloaded via the Inbox server 270 with a "metadata" report description object, reports can be presented without a report-specific presentation code. At one level, these metadata descriptions function like the catalog in a relational

database, describing each row of a result set returned from the middle tier as an ordered collection of columns. Each column has a data type, a name, and a desired display format, etc. Column descriptive information will be stored in an object, and the entire result set will be described by a list of these objects, one for each column, to allow for a standard viewer to present the result set, with labeled columns. Nesting these descriptions within one another allows for breaks and subtabling at an arbitrary number of levels. The same metadata descriptions can be used to provide common data export and report printing services. When extended to describe aggregation levels of data within reporting dimensions, it can even be used for generic rollup/drilldown spreadsheets with "just-in-time" data access.

The metadata data type may include geographic or telecommunications-specific information, e.g., states or NPAs. The report viewer may detect these data types and provide a geographic view as one of the graph/chart types.

As mentioned herein with respect to FIGS. 7 and 12, the StarODS component 400 interfaces with StarWRS web reporting tool 200 for specific customer reporting requirements. As described, the Report Requester 260 communicates with the user client 201 and controls navigation and requests for customization criteria via the Web browser. The Report Requester receives from StarOE any billing hierarchies and static pick lists needed by the client to customize report requests. Report request customizations are then passed to the Report Manager, which acts as repository of report requests, both adhoc and recurring, that are submitted for processing by the client. Along with the necessary customization criteria selected for report customization, the Report Manager 250 stores metadata about the report request, including report format information, sort, and display specifics. The Report Manager is responsible for passing report requests to the back end DSS and data marts for processing, and provides the entity against which the list of report requests known to the data marts are validated.

The Inbox server component 270 is the store and forward repository of all completed reporting requests, requests for call detail data, and any communications to the customer. As will be described, the Decision Support Server 475 ships formatted data in a compressed comma delimited format ("CDF") to the Inbox. Customers are then responsible for retrieving their report data held in the Inbox.

In accordance with the invention, the primary function of the DSS 475 is to generate priced billing report data in accordance with the customer's request. To accomplish this, the DSS interfaces with two StarWRS systems: Report Manager 250, and Inbox 270, as shown in FIG. 7. The Report Manager formats the customer's request in accordance with a defined set of rules and sends the request to the DSS. The DSS 475 reads customer's requests which are metadata descriptions of the type of priced data report requested by a customer, translates the metadata into database queries, and implements commercial off-the-shelf ("COTS") tools to run the queries against the data in the data marts, format the query results into a form readable by StarWRS report viewing components, and transmits the completed reports to the directory of the customer's Inbox, e.g., via FTP. In the preferred embodiment, Talarian Smart-Sockets™ messaging middleware is used to coordinate report requests (transmitted from the StarWRS report Manager to DSS, and report completion notification from DSS to the StarWRS Report Manager). The Report Manager formats the customer's request in accordance with a defined set of rules and sends the request to the DSS as a Talarian message with the Report Manager 250 maintaining the Talarian

Sender program, and the Decision Support Server 475 maintaining the Talarian Receiver program. Messages are sent with guaranteed message delivery ("GMD"), thus assuring all request data sent by RM is received by the DSS.

As depicted in greater detail in FIG. 13(a), a Report Manager/DSS application programming interface "API" 480 is provided whereby the RM server 250 publishes the message to the Decision Support Server in response to its receipt of a report request. Subsequently, the DSS 475 returns a "Message Received" message. When the DSS has processed the request, it publishes the message to the RM 250 with the name and location of the report file or an error message to the Report Manager, via an "NRL" metadata message as described herein.

FIG. 13(b) illustrates an DSS/Report Manager application programming interface "API" 485. In the preferred embodiment, all return messages are persistent. Thus, as shown in FIG. 7 the DSS incorporates a Talarian message queue 490 operating on a First-In-First-Out (FIFO) basis. If the DSS is unable to establish the connection with Talarian, or there is an error in transmission, the DSS queues all messages, and continues to retry until a successful send is executed.

Similarly, a DSS/Inbox API is provided to manage FTP file transmissions including: error handling, retry logic, and the ability to maintain the file name and location of where report files are stored. Particularly, the DSS/Inbox API sends the report file to the inbox (FIG. 7). If the DSS has generated an error condition, and the report is unable to be generated, an error message will be sent to the inbox in place of the report file. In either case, a return message will be delivered to the DSS/Report Manager API 485 indicating a successful or unsuccessful generation and transmission of the report file.

More particularly, as shown in FIG. 15(a), an RTServer process 377 is provided for maintaining connections, ensuring guaranteed message delivery, and tracking the success of all messaging operations. As the Report Manager interfaces with multiple systems, the RTServer 377 processes are located in the RM. The DSS is provided with RTClient processes 377a,b that provides the API to RTServer: one RTClient 377a for providing the API to Report Manager for receiving messages; and, a second RTClient 377b for providing the API for the NRL. However, it should be understood that other ODS boxes can have one RTClient. The RM and Arbitrators 360a,b use the GMD feature of Talarian to deliver messages. RM/Inbox communication is not affected by outages of ODS server as the arbitrator and ODS communication is independent of RM/Inbox communication.

In the preferred embodiment, the DSS architecture is transparent to the Report Manager which publishes Talarian messages to which the DSS will subscribe. In addition to the tokenized character string request message which specifies report type, filters, and any customer request-specific information, RM server provides additional fields as part of the Talarian request message including: a Corp_ID, Priority, and RequestID. Corp_ID allows the DSS to route the request to the appropriate data store without having to invoke a parser. Data are partitioned on Corp_ID in the ODS database warehouse. Request_ID is used to send back an ARDA failure message, in the event of an invalid message. The Priority field allows DSS to pick up the next high priority request from a queue of non-processed requests, without invoking the parser.

FIG. 14(b) illustrates the implementation of the COTS Information Advantage® Interface Object ("IAIO") 372, which is a process running in the DSS 475 for performing

the following functions: 1) publishes and subscribes Talarian messages to Report Scheduler; 2) parses the request meta-data ARD (Add Report Definition) message; 3) publishes an ARDA (Add Report Definition Acknowledgment); 4) populates a request table 390 with total, sub-total and sort information according to the received report request; 5) transforms the ARD tokens from the metadata request into an overlay file 392 which is a text file that is submitted to IA's Decision Suite™ process to generate the corresponding SQLs; 6) updates a Request Status table 391 with appropriate status, e.g., process complete, failed, in progress, etc.; and, 7) if a failure occurs, it updates an error log (not shown).

More particularly, in view of FIG. 14(b), ARD metadata request messages are received into the ODS system via arbitrator processes 360*a*,*b* which are responsible for routing the request message to the appropriate ODS database according to a Corp/ODS mapping table 365. Report Manager publishes a single message subject "Arbitrator" having the above-mentioned request, Corp_ID, and Priority field information. Report Manager uses a round robin message delivery mechanism complemented by Talarian's GMD to publish messages to the subject Arbitrator 360*a*,*b*. The arbitrator extracts the Corp_ID field from the message and maps the Corp_ID to corresponding ODS DataMart in the table 365 it maintains. The arbitrator then republishes the message with the ODS#. As shown in the FIG. 11(b), a second arbitrator process 360*b* is provided to assure fail-over capabilities.

In FIGS. 14(a) and 14(b), a Talarian receiver, referred to herein as a Talarian Interface Object ("TIO") 370, is a process that receives the Talarian message, manages the GMD functionality, and posts updates to the request table 390 and request status table 391. As shown in FIG. 14(b), the TIO receives 370 subscribe to a subject "ODS#". The receiver inserts the message received from the arbitrator into the request table 390 and request status table 391 along with the priority, timestamp and status fields. The request status table resides on the ODS database and the messages are stored in the queue to provide queuing, log and tolerance from the failures. To determine the pending messages to be processed, status field and history_stat flags are used. Appendix "J" illustrates the contents of the ODS Request table 390 and Request Status tables 391, which are part of the ODS database.

In the preferred embodiment, the tables provided in Appendix J include: an "informix_request table 390 (FIG. 14(a)) which is the table maintained for the purpose of holding specific report request information from the received ARD message, and, an "informix_req_status for holding status of DSS processes for the current request.

Thus, for the example ARD message provided in Appendix J, the request table 390 will be populated to include: a "request_id", which is the unique identifier for the request; a "msg_desc", representing a copy of the ARD message; "unique_fname", which is the unique name assigned to each request to enable tracking of individual report requests and is additionally assigned to the report returned to the report manager; a "report_dir" indicating the location of the report that Decision Suite™ generates (which may be a tab delimited report file); "format_dir" indicating the location where the report formatter generates (comma delimited file); "inbox_dir" indicating the location on the Inbox (Report Manager) where the report is sent; "inbox_size" indicating the size of the file; "entpid", indicating the Enterprise id which may consist of one or more corporate id's; "userid" which is an identifier assigned to each user of the system;

"strdtpid" which identifies each report and is similar to column id's but on the report level; "userid" which is the user-assigned identifier for a report request; "compress" having possible values '1'=yes, '2'=no indicating if a report is to be compressed, e.g., using a standard zip routine; "threshold" defining the number of lines that shall appear on the report; "totalmode" which defines how the report shall be totaled, subtotaled as indicated by possible values '0'=No total, No subtotal; '1'=Only Subtotal; '2'=Only Total; '3'=Total and Subtotal; "nrl_totals" indicating the formatter to total the columns specified in the ".hdr" file. These columns are numeric and have a subtotal flag="y" in a column id table; "format_columns" which define derived columns on which percentages are to be calculated; "error_code" for indicating parser failure or system failure. If it's a parser failure condition, the code is returned to Report Manager; "error_desc" indicating the error description; and, "rmngr_columns" which are the columns sent to the DSS by Report Manager. The formatter checks this list against the list in the .hdr file.

Similarly, the Request_Status table 391 provided in Appendix J is populated to include the status of the different processes including: "Request_Id", i.e., the unique identifier for the request, "Priority", e.g., having a value of "1", for example, meaning adhoc; a "timestamp" which is the Informix Date Time that will be used when two or more messages have same priority; and "Status" which is a char message including the following status fields: "new_message" indicating that a new message has arrived, yet to be processed; "in_IAIO" status indicating that the message is being processed by interface process IAIO; "parser_failed" status indicating an Invalid message from RM. NRL process sends a ARDA error message; "parser_success" status indicating that the message from RM is a valid message. NRL process would send a ARDA message to RM; "IAIO_complete" status indicating that the report has been generated and directory and file name fields are modified. Formatter can pick up this message; "IAIO_failed" status indicating that IA has failed to generate a report, i.e., an error has occurred generating a report; "in_formatter" status indicating that the formatter is converting the text file generated by IA to a comma delimited format. The formatter may also, if required, does the percent (%) calculations, e.g., subtotals etc.; "format_success" status indicating that the formatter successfully completed translation of the file. It also populates the inbox file name, inbox file directory, nrltotal (optional) fields in the table; "format_failed" status indicating that the formatter failed to translate the text file generated by IA; "in_fip" status indicating that the fip process is currently sending the file to inbox; "fip_success" status indicating that the file generated by formatter is fip'd to inbox; "fip_failed" status indicating that the formatted file could not be fip'd to inbox; "in_NRL" status indicating that the NRL process is trying to send either ARDA message or NRL message to RM; "NRL_sent" status and "ARDA_sent" status indicating that the respective NRL or ARDA message has been sent to RM. Each DSS process updates the request status table as it processes.

A further "history_stat" field may be provided in the request_status table 391 having a value, e.g., 'A' (Active) indicating that the record needs to be processed, or, indicating 'H' (History), when the record is no longer active and needs to be archived in a separate database set up for archival purposes (not shown).

As further shown in Appendix J, there are two more tables that are defined for DSS sorting and formatting processes: a Column ID Table, and a Translation table which are tables configured for the formatter process, as will be described.

As further shown in FIG. 14(b), in operation, each Information Advantage® Interface Object ("IAIO") 372a,b, . . . n reads the status table 391 for new entries. When a new entry is posted, it invokes a parser process 393, and invokes the Information Advantage® SQL generator engine which retrieves the requested data from the database, and updates the status table 391.

Particularly, the Decision Suite™ tool receives the overlay file (FIG. 14(a)) and performs the following functions: 1) generates SQL; 2) submits the SQL to the appropriate datamart (ODS database); 3) generates a Report file with a *.txt extension; 4) updates Request Status table 391 with appropriate status; and, 5) if a failure occurs, updates the error log. Following generation of the *.txt file, a sort process is invoked to perform the following functions: 1) reads the Request table 390 for column(s) on which to sort the Report; 2) reads the *.txt file; 3) sorts the *.txt file and generates two files: i. a file with a *.hdr extension which file contains the header information, consisting only of only column id's, and, ii. a file with a *.data extension which file contains sorted data provided in the *.txt file and is the body of the Report; 4) it further updates the request status table with a 'success' or 'failure' code; and, 5) if a failure occurs, updates the error log.

As further shown in FIG. 14(b), continuously running FTP, NRL and ARDA processes are provided to take appropriate actions in accordance with the request status table entries. For example, an FTP process 378 performs the following functions: 1) reads the status table 391 for entries ready to be sent to the Inbox and FTP's the *.csv or *.txt to the inbox 270; 2) Determines success or failure of file transfer; 3) Updates the Request Status table 391; and, if a failure occurs, updates an error log.

The NRL (Notification of Report Location) process 382 performs the following functions: 1) reads the Request Status table 391 for any success status or failure of any process; 2) Invokes a receiver process with appropriate status and file location populated in the NRL; and, 3) If failure occurs, updates the error log are text files. Particularly, should an error occur in any of the DSS processes, an error log is updated. Error log directories may be delineated by process and day of week. Each new error generated by the same process in the same day appends the log with the new message. In either event, the NRL process returns the NRL message to Report manager indicating the status and location of any generated files.

As further shown in FIG. 14(b), an ARDA process 383 reads the status table 391 for parser failures. Should the parser fail due to insufficient or missing data, ARDA process will return an ARDA message to the Report Manager with the appropriate error code. In particular, the types of conditions that result in error messages being sent to the report manager and/or local log include: i) when the request message received from the Report Manager can not be parsed due to bad data or invalid format; ii) when the SQL can not be generated due to invalid request format or parameters; iii) system or process failure; iv) cannot query database due to a database failure; etc.

For Priced Reporting, the StarWRS report requestor functionality is invoked as described in above-referenced, co-pending U.S. patent application Ser. No. 09/159,409, filed Sep. 24, 1998. Particularly, the end-to-end process 800 from a priced report request to report delivery is shown in FIG. 16(a)–16(c). Specifically, a user first establishes communication with the DMZ Web server 44 and logs on to the nMCI Interact system by entering the user's name and password onto a login dialog box. Then, an application

running on the backplane directs a "Validate User Message" common object to the StarOE server 280 via the web server and dispatcher servers (FIG. 3) to direct the StarOE server 280 to perform security validation and authenticate the user ID and password in the manner as described in commonly owned, co-pending U.S. patent application Ser. No. 09/159,514, filed Sep. 24, 1998, entitled AUTHENTICATION AND ENTITLEMENT OF WEB BASED DATA MANAGEMENT PROGRAMS, the contents and disclosure of which is incorporated by reference herein. It is understood that all communication to the StarOE server is via TCP/IP with a Unix process listening on a known TCP port. The StarOE server acts as a proxy when messages are sent from the Dispatcher server 46 and supports synchronous transactions. All data and security information is accessed by direct queries to a StarOE server database 283, such as provided by Informix. Once a user is logged on, the Web Server 44 (FIGS. 3 and 12) requests a current list of authorized applications from the StarOE server 285. Particularly, as described in co-pending U.S. patent application Ser. No. 09/159,408, filed Sep. 24, 1998, the contents and disclosure of which is incorporated by reference herein, a "Get User Application Request" message is communicated to the StarOE server via the backplane from the report requestor which queries the Informix database to obtain a list of authorized applications, i.e., services, for the user and which determines which buttons on the home page are active, thus controlling their access to products. This information is downloaded by a GUI applet that is executed via the Backplane (FIG. 4) and incorporated into the home page that is presented to the user. An exemplary home page screen display 80 is shown in FIG. 5 which provides a list of icons 70 representing the possible options available to the user according to that customer's entitlements.

Appendix H of co-pending U.S. patent application Ser. No. 09/159,409, filed Sep. 24, 1998 provides the format and content of the nMCI Interact common objects downloaded to the Report Requestor client application to enable web-based reporting. As shown in above-referenced Appendix H, the Report Requestor first asks for common objects for a user's default timezone, language and currency. The Report Requestor objects are invoked to retrieve from StarOE the various customer entitlements relating to security, geographical hierarchy, billing hierarchy, and paging and e-mail notification, as further shown in Appendix H.

In response to selection of the Report Requestor icon, a display is generated to present the reporting options to a user in accordance with that user's entitlements as previously determined. It should be understood that in the preferred embodiment, the icons for applications the user has security access to are shown bolded. Thus, for a customer subscribing to nMCI Interact Priced Reporting, a Priced Reporting icon is automatically enabled when the home page appears.

Thus, upon selection of a Report Requestor icon 76 from the home page screen display 80 of FIG. 5, a StarWRS report requestor web page is presented to the customer. The backplane object allows the user access to the Report Requestor front end if the user is so authorized, and a client priced reporting application is downloaded to the customer who is presented with the Priced reporting dialog screen (not shown), as indicated at step 802 in FIG. 16(a). It is from this screen that the user is presented with priced reporting options to view/retrieve completed reports via the StarWRS Inbox, or create a new report or, modify an existing Priced call detail data report.

Particularly, from the Priced reporting dialog screen, the user is enabled to edit an existing report maintained in the

report manager inventory, generate a new report, copy an existing report, or delete an existing report. For example, as indicated at step 805 (FIG. 16(a)), a user may initiate retrieval of the user report list containing existing user reports from the RM inventory, which process entails invoking the Report Requestor to initiate generation of a metadata request to download the report inventory from RM as indicated at step 810. The Report inventory for the specific user is loaded and displayed for the user on the user report request display screen, enabling the user to select a report, as indicated at step 812. Then, at step 815, the selected report is retrieved from StarWRS Report Manager and displayed for the customer.

Then, as indicated at steps 818 and 820, the customer may enter the desired reporting options and reporting criteria including: 1) the report product including toll-free, MCI Vision, and MCI Vnet options; 2) the report category which includes options for: analyzing traffic, call center, call detail, checking calling frequencies, financial, marketing, monitoring usage, and telecommunications categories for toll-free, Vnet and Vision customers; 3) the report type which includes priced call detail data or traffic data options; and 4) a report direction and which includes inbound, outbound, or both directions. Additionally, the user may select the report format associated with a reporting category.

Whether creating a new report or editing an existing report, the user is enabled to select customization options from successive dialog screens (not shown) that are presented to the user showing all the report customization categories for building a new report and/or editing an existing report. From this screen and related report building dialog boxes, all of the initial values for retrieving the MetaData, customization options and GUI builder options from the report manager server 250 necessary to build (edit) a report are provided in accordance with the user's entitlements. As described in greater detail in co-pending U.S. patent application Ser. No. 09/159,409, filed Sep. 24, 1998, a user may provide the following customization and report builder options: general customization options; layout customization options; access customization options; hierarchy customization options; geographic customization options; and, notification customization options.

In performing the report request process, as shown in FIG. 12, the Report Requestor client application 212 gains access to the Metadata stored at the Report Manager server 250 through messaging, as indicated at step 825. Particularly, as hereinafter described, a message generated by the Report Requestor in accordance with the user request is first received by the report manager proxy 250'. In the preferred embodiment, the report manager proxy comprises a set of tools in the form of reusable objects, preferably written in C++ code, or the like. For example, a parser object tool is employed to decompose the Metadata messages sent by the report requestor 212 to validate the message. If errors are found in the Metadata input, the RM will return an error message to the requesting client. If the Metadata passes the validation tests, the request type is then determined and the appropriate service will be invoked after which a standard response is sent back to the requesting client or and/or fulfilling server.

The Report Manager 250 implements stored procedures to translate the message, perform the request, and send the information back to the Report Requestor 212 which uses the metadata to determine what a standard report should look like, the customization options the user has, and the types of screens that should be used for the various options (i.e., single selection, multiple selections, etc.). It is under-

stood that the selection of available standard template reports is based on the user's entitlements.

The following list provides the types of requests that may be initiated by the Report Requestor 212 and the responses performed by the Report Manager 250: 1) Get/Send report template list (GRTL/SRTL)—which request retrieves the list of all standard report templates for all products and is used only to obtain general report information, e.g., report title, description, etc.; 2) Get/Send report template detail (GRTD/SRTD)—which request retrieves the details of a specific standard report template; 3) Get/Send user report list (GURL/SURL)—which request retrieves the list of all user reports for the report format selected from a user report table and is used only as a request for general report information, e.g., report title, status, etc.; 4) Get/Send user report detail (GURD/SURD)—which request retrieves the details of a specific user's report; 5) Add report definition/Acknowledgment (ARD/ARDA)—which requests addition of a user-created report to a user report table. If the report is a scheduled report, this request is also communicated to the fulfilling server at the time the report is due; 6) Delete report definition/Acknowledgment (DRD/DRDA)—which request deletes a user-created report from the user table; 7) Copy report definition/Acknowledgment (CRD/CRDA)—which request creates a duplication of the report the user is editing (other than the report title) and creates a new report ID for it; 8) Update Reporting Schedule/Acknowledgment (URS/URSA)—which request updates the scheduling information on a report without having to send a Delete and Add request; and, 9) Get Pick List/Acknowledgment (GPL/GLPA)—which request enables the Report Requestor 212 to get a pick list provided by StarOE server.

In a preferred embodiment, as shown in Table 1, the interface message sent to the RM server 250 from the report requester via the Dispatcher server 46 comprises a three to four character message acronym followed by request specific parameters.

TABLE 1

Parameter Name	Parameter Type	Required	Acceptable Value
Request	3 or 4 Characters	Yes	Msg acronym
Data parms . . .	Characters	No	

Table 2 illustrates the interface message format returned by the RM server 250.

TABLE 2

Parameter Name	Parameter Type	Required	Acceptable Value
Response Error Code	Char (4)	Yes	Msg acronym 0 = OK or error
Data parms . . .	Char #	No	

As shown in Table 2, the response message to be returned in Metadata format preferably includes a four character message acronym followed by an error code. A successful request (or a request acknowledgment) generates a response with an error code of "0". Additional data specific to the response follows this error code. If any server receives a message which is not known, the response message will echo the message acronym back along with an appropriate error code.

Appendix A provides a series of tables containing the content for each metadata message request that can be sent by the report requestor 212 for each of the enumerated user requests, in addition to the content of the corresponding metadata message responses by the RM server 250. As an example, when a user requests a list of all standard report templates that can be created for a specified product, category, and product type, e.g., toll free unpriced data, an example metadata format is as follows:

```
GURL=PRODUCT=V,DATATYPE=R,DATACAT=P,IO=O>
```

where GRIL is the message name, the PRODUCT indicates the product type, e.g., V=Vnet, C=CVNS, S=Vision, T=toll free, F=Traffic view, etc. DATATYPE indicates the data type, e.g. R=reports, D=call detail, etc., and DATACAT represents the report category, e.g., P=priced, U=unpriced.

In the hereinafter described manner, the GRIL message is received by the StarWRS proxy server application 250¹ to enable the RM server 250 to perform the query into the RM Informix database having the data associated with the request. Specifically, after selecting the Report Requester from the browser or the Toolbar, a WRSApp object is launched. At its creation, the WRSApp object creates a DataManager object to guide the data and which initiates a CommunicationManager object to manage all communication between the client and the server. The CommunicationManager utilizes a RptManagerMsg object to create: 1) a GRIL; 2) a WRSCommWrapper for direct communication with the backend; and, 3) a WRSReportManagerUtilParser to format the data returned. In response, the Report Manager creates a Dispatcher object, which contains the business logic for handling metadata messages at the back-end and utilizes the services of a RMPParser class. Upon determining that the client has sent a valid message, the appropriate member function is invoked to service the request. Upon receiving the message, the Report Manager creates the Parser object (RMPParser) which takes the message apart and invokes a validation object which validates the message.

In response to the GRIL message, the data returned by the Report Manager server 250 for this particular request may include the following data in metadata format as follows:

```
SURL=ERROR=0, REPORTS = <RptCategoryDescription1>
<RptTitle1.1, RptTemplateID1.1, RptCategoryType1.1>,
<RptTitle1.2, RptTemplateID1.2, RptCategoryType1.2>,
<RptCategoryDescription2>=<RptTitle2.1,
RptTemplateID2.1, RptCategoryType2.1>, <RptTitle2.2,
RptTemplateID2.2, RptCategoryType2.2>, ...
<RptCategoryDescription#>=<RptTitle#n.1,
RptTemplateID#n.1, RptCategoryType#n.1>, <RptTitle#n.n,
RptTemplateID#n.n, RptCategoryType#n.n>>
```

wherein RptID# indicates a standard report template ID, RptTitle# indicates the standard report template title, RptCategory# indicates the report category, e.g. Monitor Usage, Analysis Traffic, Historical, Executive Summary, Call Detail, etc.; and, RptDescript indicates the standard report template description displayed to the user. Thus, for each Report Template Category, there will be the list of reports with each entry containing a Report Template Title, a Report Template Description and the Report Template ID.

The SRTL message is sent from the StarWRS RM proxy server to the report requestor for presentation to the customer. Specifically, the SRTL response is built inside the

esql wrapper function after obtaining the necessary information through the stored procedure from the Report Manager Informix database. The Report Manager creates the RMServerSocket object and sends the SRTL message back to the client.

To retrieve details of the standard report template, the GRITD request message request is sent having content shown in the table in Appendix A. When specified, the Report ID field indicates an existing report that a user may wish to edit.

The SRTD response generated by the RM server is formatted in metadata as follows:

```
< Report Template ID=ID#,
NODE1=<node level1, label value1, assigned unique screen
identification1, >,
NODE2=<node level2, label value2, assigned unique screen
identification2, >control ID2.1, field value2.1, data
location2.1>, <control ID2.2, field value2.2, data
location2.2>, <...>>,
NODE#n=<node level#n, label value#n, assigned unique
screen identification#n, >control ID#n.1, field
value#n.1, data location#n.1>, <control ID#n.2, field
value#n.2, data location#n.2>>
```

In the SRTD message, the MetaTreeData Label fields include such values as General, Report Name, Report Description, Scheduled Execution, etc. The MetaCtrlInfo MetaField Value fields may be blank or may contain the selection options available to the user. This information is taken from the report template database.

As another example, when a report request is submitted to retrieve a full list of user created reports from a user report table, i.e., a template list for a particular report product, category, and type, the example metadata format is as follows:

```
GURL<USERID=jeanvnc2,RPTTMPID=1,ENTPID=00029224,
PRODUCT=T,DATACAT=U>
```

with UserID and ReportTemplateID fields specified. Specifically, this process entails invoking the CommunicationManager object to communicate with the RM server in order to obtain a SURL metadata message. The CommunicationManager utilizes the RptManagerMsg object to create: 1) a GURL, 2) a WRSCommWrapper for direct communication with the backend, and, 3) a WRSReportManagerUtilParser to format the data returned. The parser returns a hash table containing the User Report List. At the RM server, the Report Manager creates a Dispatcher object that contains the business logic for handling metadata messages at the back-end and utilizes the services of the RMPParser class. Upon determining that the client has sent a valid message, the appropriate member function is invoked to service the request. The Report Manager, upon receiving a message, creates a Parser object (RMPParser) which takes the message apart and invokes a validation object which validates the message.

In response to the GURL request, the data returned is taken from a user report table in the RM server database. The generic SURL message in Metadata format returned by the RM server 250 includes the following information:

```
REPORTS = <UserRptCategory1 = <UserRptTitle1,
UserRptID1, activeflag, report type, statusdate >>,
<UserRptCategory2 = <UserRptTitle2, UserRptID2,
activeflag, report type, statusdate>>, ...
<UserRptCategory#n = <UserRptTitle#n, UserRptID#n,
activeflag, report type1 statusdate>>>
```

wherein for each user report category, there is a list of reports where each entry contains a UserRptID# indicating a user-defined report template ID, a UserRptTitle# indicating the user's report template title, and a UserRptCategory# indicating the user report category. Specifically, the SURL response is built inside an esql wrapper function after obtaining the necessary information through a stored procedure from the Informix database. The Report Manager creates the RMServerSocket object and sends the SURL message back to the client.

To retrieve the details of a specific user's report, the GURD message is sent having data as contained in the table shown in Appendix A. Specifically, when the user selects a report from the Inventory List on the Report Requestor, a Communication Manager object is invoked to communicate with the RM server in order to obtain a SURL metadata message. The CommunicationManager object first utilizes the RptManagerMsg object to create: 1) a GURD metadata message, 2) a WRSCCommWrapper for direct communication with the backend, and 3) the RSReportManagerUtil-Parser to format the data returned. The parser organizes the data into a series of nodes which are utilized to create the report builder tree on the report requester customization screen. Later this data will be extracted from the node and

```
< Report Template ID=ID#,
NODE1=<node level1, label value1, assigned unique screen
identification1, >,
NODE2=<node level2, label value2, assigned unique screen
identification2, <control ID2.1, field value2.1, data
location2.1>, <control ID2.2, field value2.2, data
location2.2>, <...>>,
NODE#n=<node level#n, label value#n, assigned unique
screen identification#n, <control ID#n.1, field
value#n.1, data location#n.1>, <control ID#n.2, field
value#n.2, data location#n.2>, <...>>>
```

This response thus may include the report information having detailed items including: UserReportID (UserID), User's report name (UserName), product (UserProd), Threshold (UserThreshold), User Report Description (UserDescript), Report Columns (UserFields), Report column headings (UserHeaders), and, in addition, customization options with fields indicating, inter alia, columns to display (UserHeaders), user-defined criteria (UserCriteria), a sort order (UserOrder) and scheduling selections (UserSched), the last update of this report (UserLastUpdate) and, the Report status (if ad hoc) (UserStatus), etc.

If a request is made to add a user-created report to a User_report table maintained by the RM Server 250 and the RS server 260, the ARD metadata message having fields defined in the table provided in Appendix A is processed by the RM server 250, as indicated at step 828, FIG. 16(a). An example message in metadata format to initiate the addition of a user-created report for ODS (Inbound/Outbound) reporting data is as follows:

```
ARD<USERID=<janvat2.ENTPID=00022924,STDRPTID=00,NAME=<Ci
ty Summary Outbound,PRODUCT=S,CATEGORY=Analyze Traffic,
THRESHOLD=<RECCOUNT=20>,SCHEDULE=A<START=199806020000,EN
D=199807151200>,RANGETYPE=1,SCHEDIYPE=A,TIMEZONE=45,BILL
ING=<INBOUND<=<00000013,00000013><NANA><NANA><INBOUND=<0
0000004,00000004><NANA><NANA><NANA><CARDNO=<654654>~546546
46546546>,IDAC=<654654>~1246>,GEO=<GEO<001,001
USA/WORLZONE1><NANA><NANA><NANA><NANA><GEO<001,001
USA/WORLZONE1><O,CO><NANA><NANA><NANA><OACCESS=<4-
1>,ODISTRANGE=<A>F>,OUSAAG=<4>~4>,SORTBY=<4D>~4D>,DESSCRIPTIO
N=<This report summarizes call detail by the terminating
city and state (USA) / province (CA). The report is
based on the date/time ranges and report criteria
selected>,COLUMNS=<54-55-67-62-36-61-58-63-66-66>>ACT
IVE=t,TOTALMODE=0,EMAIL=0,PAGE=0, LANG=1234, CURR=2345>
```

used to construct the screen related to the node. The Report Manager server creates the MCDISPATCHER object which contains the business logic for handling metadata messages at the back-end and utilizes the services of the RMParser class. Upon determining that the client has sent a valid message, the appropriate member function is invoked to service the request. The Report Manager, upon receiving a message, creates the Parser object (RMParser) which takes the message apart, invokes a validation object which validates the message and builds a response inside the esql wrapper function after obtaining the necessary information through the stored procedure from the Informix database. The Report Manager creates the RMServerSocket object and sends the SURL.SRDT message back to the client. The responsive SURL metadata message corresponding to a retrieve user report detail (GURD) request has the following metadata syntax:

In this example, the "NAME" field refers to the Report Name (e.g., city summary); the "PRODUCT" field refers to the report product (Vision); the "THRESHOLD" field refers to the record count; the "DESCRIPTION" field refers to the report description; the "COLUMNS" refers to the number of columns specified for a report by the user; the "BILLING" field refers to the specified report billing entitlement, i.e., billing hierarchy; the "ACCESS" field refers to the inbound access type and the "OACCESS" refers to the outbound access; the "SORTBY" field indicates the report column sorting customization with "A" indicating column(s) having data to be sorted in ascending order and, "D" indicating column(s) having data to be sorted in descending order; the "SCHEDULE" field referring to the scheduling type, e.g., with "A" indicating an ad-hoc report, and the user specified date range on which to report as indicated by the "START" and "END" fields, and additionally, the scheduling fre-

quency information in the case of a recurring report; the SUBTOTALCOLUMNS field, referring to the report columns having data to be subtotaled; and, the "EMAIL" and "PAGE" fields indicating reporting notification via e-mail or paging, respectively.

Furthermore, for each of the metadata messages in Appendix A, including the Delete Report Definition (DRD), copy report definition (CRD), and update report scheduling (URS) messages, the report manager server 250 responds to the Report Requestor with the processing results. In the case of a copy report, a new User Report ID is assigned and returned by RM. When editing an existing StarODS (priced call data) report, the user may make changes to the Report Title, the Report Description, the Report scheduling, the 800 numbers and thresholds, and may customize number of rows, report columns, access codes, access types, billing location, geographic location, paging notification, and e-mail notification. More specifically, when the user selects a report from the inventory list or a new report, an WRSedit Screen is launched to provide the editing capabilities which are available for the report format. WRSedit guides the screens through the process of retrieving the screens' data. Some of the screens need data which has not yet been retrieved, such as 800 numbers or geographic locations. These screens manage the requests to the DataManager object to create the get pick list (GPL) message (Appendix A), which launches the CommunicationManager object to perform this task. The CommunicationManager utilizes the RptManagerMsg object to create the GPL, the WRSCommWrapper for direct communication with the backend, and the WRSReportManagerUtilParser to format the data returned. In response, the Report Manager server creates the MCIDispatcher object and invokes the MCIRMPParser class. Upon determining that the client has sent a valid message, the appropriate member function is invoked to service the request. The Report Manager, upon receiving a message, creates the Parser object (RMPParser) which takes the message apart and a validation object is invoked which validates the message. The response is built inside the esql wrapper function after obtaining the necessary information through the stored procedure from the Informix database. The Report Manager creates the RMServerSocket object and sends the GPLA message back to the client.

Having described the functionality of selecting and/or generating a report and customizing it, reference is now had to the process for running the report request in StarODS. Particularly, in the preferred embodiment, the user may select a save and exit report option, or a save and run report option. In either scenario, an WRSedit object enables a WRSScnMgr object to save the report to the RM server. The WRSScnMgr object launches each screens save method which communicates with the DataManager object to place the screens data in its corresponding WRSNode. Once all of the WRSNode objects have been updated, the WRSScnMgr object calls the DataManager object's SaveReport method to build a hash table to contain all of the report's data. The CommunicationManager utilizes the RptManagerMsg object to create the ARD metadata message from the hash table, the WRSCommWrapper for direct communication with the backend, and the WRSReportManagerUtilParser to handle any errors thrown by the server. The Report Manager creates the Dispatcher object, and utilizes the services of the RMPParser class and validation objects. Upon determining that the client has sent a valid message, the appropriate member function is invoked to service the request. The response is built inside the esql wrapper function after obtaining the necessary information through the stored pro-

cedure from the RM database. The Report Manager creates the RMServerSocket object and sends the ARDA message back to the client.

As illustrated in FIG. 16(a), at step 830, in reference to user selection of a Save and Run report option, the report is marked as scheduled and saved in a user_table in the Report Scheduler server 260 via the Report Manager. Subsequently, as indicated at step 630, the Report Scheduler server 260 generates an ARD message (Appendix D) and sends the ARD message to StarODS DSS server for which the DSS has a predefined interface, as described herein.

Next, as indicated at step 832, the DSS receives the request and acknowledges receipt. Specifically, when the request is received it is first validated with StarOE to ensure that the user is entitled to receive information about the selected product corp and number(s). Once the request passes validation, the DSS IAIO reads the header to determine which Data Mart will ultimately be queried. It then parses the metadata into a format which the COTS software can readily convert into a SQL statement, as indicated at step 835, FIG. 16(b), and adds the report to the DSS report queue based upon type (Daily, Weekly, Monthly, Adhoc) and associated DataMart, as indicated at step 638. It should be understood that at this point, the request has been flagged as submitted in the RM database, as indicated at step 633.

From this point forward, DSS activity is controlled by a control process and progress or errors are logged internally in the DSS system. This control process includes logic enabling the prioritization of report requests and application of rules defining the order in which they should be executed. Thus, at the appropriate time, depending on the type or report, reporting period and other parameters, the Information Advantage query engine selects the report from the queue, as indicated at step 840, which action is logged in the report status table (Appendix J) as indicated at step 842. The SQL statement is then built by Decision Suite™ and routed to the appropriate data mart for execution in the manner as described herein, as indicated at step 843. The query engine generates the SQL statement from the metadata and executes the report which action is logged in the report status table as indicated at step 845. Next, as indicated at step 848, the query results are returned, and, a post-SQL formatting process is invoked.

More particularly, as shown in FIG. 15(b), a Formatter module 395 may perform various report result transformations including: 1) Converting of column headers generated by Information Advantages into appropriate column ids that are recognizable to the StarWRS client viewer functionality (as indicated at step 850, FIG. 16(b)); 2) Provide subtotalling for specific requested "subtotal by" columns in the format required by the StarWRS client interface (as indicated at step 853, (FIG. 16(b))) and provides report-based totals as requested by customer; 3) converting binary stream data file to ASCII text file (as indicated at step 855, FIG. 16(c)); 4) implementing Replace logic, e.g., replacement of "TAB" delimiters with appropriate "comma" field delimiters (as indicated at step 857 FIG. 16(c)); 5) implementing Repeat/Padding logic, i.e., identifying compressed columns/values and decompressing (or repeating) the values that were compressed; 6) providing alphanumeric translations for any encoded data elements returned in the result set data file (as indicated at step 859, FIG. 16(c)); and, 7) adding new computed/derived columns, e.g., percents, averages of column data values, etc., as requested by customers on specific reports.

Particularly, as shown in FIG. 15(b), the Formatter process 395 reads the *.hdr files and *.data files from the

Decision Suite™ result set to obtain respective column names and report data. Particularly, the formatter process for converting Column Headers from Information Advantage® column header names to column ids implements a lookup of column ids in a column_id's table, shown in Appendix J, based on column header names.

Then, the formatter process reads the request table 390 for total/subtotal, threshold, etc. information associated with the current report request and determines any other formatting features to be enabled for a particular result set. As shown in the example Request Table of Appendix J, parameters passed to the formatter module indicate any report request specific details that are required by the Formatter. For example, for report totals, a "total-mode" variable is used to indicate if report totals and/or sub-totals should be included. Particularly, Column IDs representing the data columns upon which subtotaling is based are passed as parameters to the Formatter process 395 and are referred to as "Break Columns". At appropriate changes in values for these break columns, the formatter generates a subtotal line for subtotaling the applicable additive facts including, for example, Call Amount, Call Duration, and Call Count.

Furthermore, the formatter reads a Column id table 396 (detailed in Appendix J) to determine data types and if any data translations are needed.

As computed/derived columns may be included or excluded from customer report requests, the Formatter process 395 for calculating new computed/derived columns on specific customer-requested reports are provided on a report request basis. Example types of derived columns include: 1) Percents, e.g., based on the additive data facts pertinent to the report request and are typically based on report totals and row amounts for Call Amount, Call duration, and Call Count; 2) Row-wise derived data elements as requested, which represent data elements computed based on original additive data elements on a row by row basis (i.e., column x/column y for each row in the result data file) and typically include average calculations such as Average # of Minutes per Call, Average Amount per Call, and Average Amount per Minute. Appendix J illustrates a derived column "percent" calculation indicated in the Column ID table showing an equation for calculating a value of a particular value (C36) divided by a column total (CT36)×100.

The Formatter process 395 may additionally perform alphanumeric translations for any encoded data elements returned in the result set data file by implementing appropriate lookup in a Translation table 397, such as the example Translation Table provided in Appendix J, and replacing the code.

Referring back to FIG. 16(c), after formatting the report, as indicated at step 860, a message is sent to the control process to update the request status table 391. It should be understood that, if a failure occurs during formatting, the error log is updated and a status message sent to the request

status table 391, as well. Then, as indicated at step 865 (FIG. 16(c)), the formatter 395 creates a *.csv (Comma Separated Value) or .txt file, gives the file a unique name and saves the file. Preferably, a *.csv is the file generated if the report is successfully generated.

As indicated at step 868, the *.csv report/data file is then "pushed", implementing FTP, to the StarODS server's directory on the Inbox server 270. The StarODS server 400 is responsible for generating unique file names within their directory on the Inbox server 270. For example, the following directory and file naming conventions used for reports generated by the StarODS server are labeled inbox/files/ods with text files having the suffix *.txt or *.txt.zip (compressed), and comma separated files having a suffix *.csv or *.csv.zip (compressed).

Finally, as indicated at step 870, once the file has been successfully transferred to the Priced reporting directory on the Inbox server, and the request status table 391 appropriately updated at step 875, the NRL process (FIG. 14(b)) generates and transmits an NRL message to the RM Server 250 notifying it of the report file name and location in the Inbox, requester information, and if the transfer was successful. This is accomplished by using an "NRL" metadata message.

Appendix B provides a table comprising the Notify Report Location parameters used for the NRL Metadata messaging sent by StarODS fulfilling server to the RM Server 250 when a requested report is complete. An example NRL message sent from the ODS server 400 to the RM server 250 is as follows:

```
NRL<TYPE=Sim-Msg-40,ENTPID=00022924,USERID=dornd,
SIDRPTID=41,USERRPTID=3415,REQUESTID=20341,COMPRESS=0,
LOC=inbox/files/testODS/STRPTID41STM_082598_084920.CSV,
FSIZE=389,PRESORTED=0>
```

An NRLA response is sent back to the DSS as shown in Appendix B.

Once the RM server 250 has received the NRL message from the fulfilling server, it verifies the file's presence and builds a metadata file, e.g., by compressing the appropriate metadata (for displaying the report) into a .MTD file. This .MTD file is utilized by the Report Viewer to know how to display the report. The Report Manager server creates a file including the metadata using the same file name as the report/data file, but having the following suffix: *.mtd or *.mtd.zip indicating a metadata or compressed metadata file, respectively.

Appendix F details the parameters that are passed in the GET METADATA messaging for indicating to the Report Viewer how to display a requested report. For example, a GET METADATA message corresponding to an Priced TVS fulfilling server report is as follows:

```
<METADATA><CRITERIA=<Name=UsageSummary2?>A>Description=
This report summarizes calls based on call type. A
Report_Level=<NBOUND=<00000001,90000001>><NA>NA><NA>NA>>
NBOUND=<00000002,90000002>><A><A>> AOptions=AScheduling
Information=AOne Time=ADates=<06/01/199800:30>-<07/01/
199800:00> ATimezone=ESTLang=1234 Curr=2345>DEFAULT_GRAPH
H_MODE=0 ADEFAULT_GRAPH_TYPE=0 ADEFAULT_X_AXIS=0
AX_AXIS_COLUMN= ADEFAULT_Y_AXIS_COLUMN=<A>
COLUMN_DISPLAY_ORDER=<105 A114 A67 A62 A36 A61 A58 A63 A6
4 A66 A65> ASORT_ALLOWED=1 APRESORTED=0 A
```

-continued

```

PRESUBTOTAL=0 '1' ATOTALMODE=0 'ASORT_COLUMN S=<105A>' A
SUBTOTAL_COLUMNS=<> ANSELC(H) SECTION=0 A
METACOLUMN=<META_COLUMN_ID=0> A
COLUMN_LABEL=Usage Description ADATATYPE=S 'ADECTMAL=0' A
HIDEABLE=1 AGRAPHABLE=0 AWIDTH=20 ACALCULATE=0 A
CALCULATE_EXPRESSION=> AMETACOLUMN=<META_COLUMN_ID=114 A
COLUMN_LABEL=Range Distance Description ADATATYPE=S ADECTM
AL=0 AHIDEABLE=1 AGRAPHABLE=0 AWIDTH=20 ACALCULATE=0 A
CALCULATE_EXPRESSION=> AMETACOLUMN=<META_COLUMN_ID=67 A
COLUMN_1 ARFI=Cals ADATATYPE=A ADECTMAL=0 AHIDEABLE=1 A
GRAPHABLE=1 AWIDTH=7 ACALCULATE=0 ACALCULATE_EXPRESSION=>
AMETACOLUMN=<META_COLUMN_ID=62 ACOLUMN_LABEL=% Cals A
DATATYPE=N ADECIMAL=1 AHIDEABLE=1 AGRAPHABLE=1 AWIDTH=7 A
CALCULATE=0 ACALCULATE_EXPRESSION=> A
METACOLUMN=<META_COLUMN_ID=35 ACOLUMN_LABEL=Minutes A
DATATYPE=N ADECIMAL=1 AHIDEABLE=1 AGRAPHABLE=1 AWIDTH=6 A
CALCULATE=0 ACALCULATE_EXPRESSION=> A
METACOLUMN=<META_COLUMN_ID=60 ACOLUMN_LABEL=% Min A
DATATYPE=N ADECIMAL=1 AHIDEABLE=1 AGRAPHABLE=1 A
WIDTH=5 ACALCULATE=0 ACALCULATE_EXPRESSION=> A
METACOLUMN=<META_COLUMN_ID=58 ACOLUMN_LABEL=Amount ADATAT
YPE=C ADECIMAL=2 AHIDEABLE=1 A
GRAPHABLE=1 AWIDTH=7 ACALCULATE=0 ACALCULATE_EXPRESSION=>
AMETACOLUMN=<META_COLUMN_ID=63 ACOLUMN_LABEL=% Amt A
DATATYPE=N ADECIMAL=1 AHIDEABLE=1 AGRAPHABLE=1 AWIDTH=5 A
CALCULATE=0 ACALCULATE_EXPRESSION=> A
METACOLUMN=<META_COLUMN_ID=64 ACOLUMN_LABEL=Avg Min/Call
ADATATYPE=N ADECIMAL=2 AHIDEABLE=1 AGRAPHABLE=1 A
WIDTH=12 ACALCULATE=0 ACALCULATE_EXPRESSION=> A
METACOLUMN=<META_COLUMN_ID=66 ACOLUMN_LABEL=Avg
Amt/Call A
DATATYPE=N ADECIMAL=2 AHIDEABLE=1 AGRAPHABLE=1 AWIDTH=12
ACALCULATE=0 ACALCULATE_EXPRESSION=> A
METACOLUMN=<META_COLUMN_ID=65 ACOLUMN_LABEL=Avg Amt/Min A
DATATYPE=N ADECIMAL=2 AHIDEABLE=1 AGRAPHABLE=1 A
WIDTH=11 ACALCULATE=0 ACALCULATE_EXPRESSION=>>>
*METADATA= <CRITERIA> <Name=My Report, Total=Totals are
located at the bottom of the report, Description=My
report description, Number_Dialed=<800>1, 800, 800>,
Scheduling_Information=Recurring, Dates= Monthly>
DEFAULT_GRAPH_MODE=1, DEFAULT_GRAPH_TYPE=1,
DEFINE_X_AXIS=1, X_AXIS_COLUMN=2,
DEFAULT_Y_COLUMNS=<5,6>,
COLUMN_DISPLAY_ORDER=<1,2,3,4,5,6>,
COLUMN_STORED_ORDER=<4,3,2,5,6,1>, SORT_ALLOWED=1,
PRESORTED = 1, TOTALMODE=3, SUBTOTALCOL=<5,6>, SELECTED
SECTION=1, METACOLVN=<META_COLUMN_ID=1,
COLUMN_LABEL=mtc, DATATYPE=S, DECIMAL=0, HIDEABLE=1,
GRAPHABLE=0, WIDTH=10, CALCULATE=1,
CALCULATE_EXPRESSION=<1 / 7>>>>

```

Once the metadata file corresponding to the requested report is build by the Report Manager, the RM flip's the .MTD file to the Inbox server. The RM server additionally updates a User report table status field with a status "C" indicating completion.

Once the Report Manager has updated the status field, the RM server 250 then adds the report to the user's Inbox.

Appendix C provides a table showing the fields for the metadata messaging between the RM server 250 and the Inbox server 270 for adding an item into the StarWRS system Inbox server 270, and the respective acknowledgment message format back from the Inbox server. In the "A" message found in Appendix C, the "LOC" field includes information about where the report data is located. For example, a metadata message indicating to the Inbox server that a priced ODS server report is available is shown as:

45

```

A<CATEGORY=R,TYPE=mtfc,REQUESTID=32197,USERID=
1>mtfc1evy2,RPTID=150,PRIORITY=COMPRESS=0,UNOTIFY=
0,MMADDR=MMTEXT=,PGT=,PGPIN=,PGTTEXT=,RPTCATEGORY=
Service Location & Hour,
LOC=/inbox/files/ods/902512294STDRPTID30.CSVENTP
ID=10324488,RQSTDT=1998 01 02
15:18,FSIZE=3705,RPTTITLE=Summary by Service
Location and Hour,MSIZE=332>

```

55

Particularly, the RM server supplies a metadata "A" message to the Inbox indicating the FTP file location. Via the report viewer, the report is now available for viewing, downloading, saving, or printing by the user, and as described in further detail in co-pending U.S. patent application Ser. No. 09/159,512, filed Sep. 24, 1998, entitled MULTI-THREADED WEB BASED IN-BOX FOR REPORT MANAGEMENT, the contents and disclosure of which are incorporated by reference as if fully set forth herein. Particularly, as shown in the exemplary nMCI home page in FIG. 4, the nMCI Interact Message Center icon 77 may be selected which will cause the display of a web page

60

65

including the message center dialog window. From the message center dialog window, a user may select from among three tabs, one of which, a reports tab, enables the retrieval of both a data file and a metadata file from the Inbox Server corresponding to those reports that have been run and available for customer viewing. Information provided for display by the message center display 325 is provided by the User table which keeps track of the status of all reports for a particular user. By double-clicking a chosen report, a report viewer application is enabled to display the chosen report on a web-page. To view the report the user selects the report and, the report metadata and the appropriate viewer are uploaded to the user (client) workstation.

As mentioned herein with respect to FIG. 3, the messages created by the client Java software are transmitted to the StarWeb (DMZ) Server 44 over HTTPS. For incoming (client-to-server) communications, the DMZ Web servers 44 decrypt a request, authenticate and verify the session information. The logical message format from the client to the Web server is shown as follows:

```
|| TCP/IP || encryption || http || web header ||
dispatcher header || proxy-specific data ||
```

where “||” separates a logical protocol level, and protocols nested from left to right. FIG. 17 illustrates a specific message sent from the client browser to the desired middle tier server for the particular application. As shown in FIG. 17, the client message 340 includes an SSL encryption header 342 and a network-level protocol HTTP/POST header 344 which are decrypted by the DMZ StarWeb Server(s) 44 to access the underlying message; a DMZ Web header 346 which is used to generate a cookie 341 and transaction type identifier 343 for managing the client/server session; a dispatcher header 345 which includes the target proxy identifier 350 associated with the particular type of transaction requested; proxy specific data 355 including the application specific metadata utilized by the target proxy to form the particular messages for the particular middle tier server providing a service; and, the network-level HTTP/POST trailer 361 and encryption trailer 366 which are also decrypted by the DMZ Web server layer 44.

After establishing that the request has come from a valid user and mapping the request to its associated session, the request is then forwarded through the firewall 550 over a socket connection 33 to one or more decode/dispatch servers 46 located within the corporate Intranet 60. The messaging sent to the Dispatcher will include the user identifier and session information, the target proxy identifier, and the proxy specific data. The decode/dispatch server 46 authenticates the user's access to the desired middle-tier service.

As shown in FIG. 17, the StarWeb server forwards the Dispatcher header and proxy-specific data to the Dispatcher, “enriched” with the identity of the user (and any other session-related information) as provided by the session data/cookie mapping, the target proxy identifier and the proxy-specific data. The dispatch server 46 receives the requests forwarded by the Web server(s) 44 and dispatches them to the appropriate application server proxies. Particularly, as explained generally above with respect to FIG. 12, the dispatch server 46 receives request messages forwarded by the DMZ Web servers and dispatches them to the appropriate server proxies. The message wrappers are examined, revealing the user and the target middle-tier service for the request. A first-level validation is performed, making sure that the user is entitled to communicate with the

desired service. The user's entitlements in this regard are fetched by the dispatch server from Order Entry server 280 at logon time and cached. Assuming that the Requestor is authorized to communicate with the target service, the message is then forwarded to the desired service's proxy, which, in the accordance with the principles described herein, comprises: 1) a report manager proxy 250 corresponding to the RM Server 250, 2) a report scheduler proxy 260 corresponding to the RS Server 260, and 3) an inbox server proxy 270 corresponding to the Inbox Server 270. Each of these proxy processes further performs: a validation process for examining incoming requests and confirming that they include validly formatted messages for the service with acceptable parameters; a translation process for translating a message into an underlying message or networking protocol; and, a management process for managing the communication of the specific customer request with the middle-tier server to actually get the request serviced. Data returned from the middle-tier server is translated back to client format, if necessary, and returned to the dispatch server as a response to the request.

FIGS. 18(a) and 18(b) are schematic illustrations showing the message format passed between the Dispatcher 46 and the application specific proxy (FIG. 18(a)) and the message format passed between the application specific proxy back to the Dispatcher 46 (FIG. 18(b)). As shown in FIG. 18(a), all messages between the Dispatcher and the Proxies, in both directions, begin with a common header 110 to allow leverage of common code for processing the messages. A first portion of the header includes the protocol version 115 which may comprise a byte of data for identifying version control for the protocol, i.e., the message format itself, and is intended to prevent undesired mismatches in versions of the dispatcher and proxies. The next portion includes the message length 120 which, preferably, is a 32-bit integer providing the total length of the message including all headers. Next is the echoing flag portion 122 that is intended to support a connectivity test for the dispatcher-proxy connection. For example, when this flag is non-zero, the proxy immediately replies with an echo of the supplied header. There should be no attempt to connect to processes outside the proxy, e.g. the back-end application services. The next portion indicates the Session key 125 which is the unique session key or “cookie” provided by the Web browser and used to uniquely identify the session at the browser. As described above, since the communications middleware is capable of supporting four types of transport mechanisms, the next portion of the common protocol header indicates the message type/mechanism 130 which may be one of four values indicating one of the following four message mechanisms and types: 1) Synchronous transaction, e.g., a binary 0; 2) Asynchronous request, e.g., a binary 1; 3) Asynchronous poll/reply, e.g., a binary 2; 4) bulk transfer, e.g., a binary 3.

Additionally, the common protocol header section includes an indication of dispatcher-assigned serial number 135 that is unique across all dispatcher processes and needs to be coordinated across processes (like the Web cookie (see above)), and, further, is used to allow for failover and process migration and enable multiplexing control between the proxies and dispatcher, if desired. A field 140 indicates the status is unused in the request header but is used in the response header to indicate the success or failure of the requested transaction. More complete error data will be included in the specific error message returned. The status field 140 is included to maintain consistency between requests and replies. As shown in FIG. 18(a), the proxy

specific messages 375 are the metadata message requests from the report requester client and can be transmitted via synchronous, asynchronous or bulk transfer mechanisms. Likewise, the proxy specific responses are metadata response messages 380 again, capable of being transmitted via a synch, asynch or bulk transfer transport mechanism.

It should be understood that the application server proxies can either reside on the dispatch server 46 itself, or, preferably, can be resident on the middle-tier application server, i.e., the dispatcher front end code can locate proxies resident on other servers.

As mentioned, the proxy validation process includes parsing incoming requests, analyzing them, and confirming that they include validly formatted messages for the service with acceptable parameters. If necessary, the message is translated into an underlying message or networking protocol. A list of Report Manager and Inbox proxy error messages can be found in Appendix E. If no errors are found, the proxy then manages the communication with the middle-tier server to actually get the request serviced. The application proxy supports application specific translation and communication with the back-end application server for both the Web Server (java applet originated) messages and application server messages.

Particularly, in performing the verification, translation and communication functions, the Report Manager server, the Report Scheduler server and Inbox server proxies each employ front end proxy C++ objects and components. For instance, a utility program and a C++ components library, is provided for implementing general functions/objects. Various C++ parser objects are invoked which are part of an object class used as a repository for the RM metadata and parses the string it receives. The class has a build member function which reads the string which contains the data to store. After a message is received, the parser object is created in the RMDispatcher object which is file containing the business logic for handling metadata messages at the back-end. It uses the services of an RMParse class. Upon determining that the client has sent a valid message, the appropriate member function is invoked to service the request. Invocation occurs in MCIRMServerSocket.C when an incoming message is received and is determined not to be a talaian message. RMSEIverSocket.C is a class implementing the message management feature in the Report Manager server. Public inheritance is from MCIServerSocket in order to create a specific instance of this object. This object is created in the main loop and is called when a message needs to be sent and received; a Socket.C class implementing client type sockets under Unix using, e.g., TCP/IP or TCP/UDP. Socket.C is inherited by ClientSocket.C; Socket (theSocketType, thePortNum) and ServerSocket.C; Socket (theSocketType, thePortNum) when ClientSocket or ServerSocket is created. A ServerSocket.C class implements client type sockets under Unix using either TCP/IP or TCP/UDP. ServerSocket.C is inherited by RMServerSocket when RMServerSocket is created. An InboxParser.C class used as a repository for the RM Metadata. The class "build" member function reads the string which contains the data to store and the class parses the string it receives. After a message has been received, the MCInboxParser object is created in inboxutil.C which is a file containing the functions which process the Inbox requests, i.e., Delete, List, Fetch and Update (Appendix G). Additional objects/classes include: Environ.C which provides access to a UNIX environment; Process.C which provides a mechanism to spawn slave processes in the UNIX environment; Daemon.C for enabling a process to become a daemon; Exception.C for exception

handling in C++ programs; and, RMLog.C for facilitating RM logging. In addition custom ESQ.C code for RM/database interface is provided which includes the ESQ.C interface (Informix) stored procedures for performing the ARD, DRD, DUR, URS, GRD, CRD, and GPL messages. The functions call the stored procedures according to the message, and the response is build inside the functions depending on the returned values of the stored procedures. A mainsql.C program provides the ESQ.C interface for messages from the report manager and report viewer.

A list of Report Manager and Inbox proxy error messages can be found in Appendix E.

Outgoing (server-to-client) communications follow the reverse route, i.e., the proxies will feed responses to the decode/dispatch server, which will encrypt the client-bound messages and communicate them to the DMZ Web servers over the socket connection. The Web servers will forward the information to the client using SSL. The logical message format returned to the client from the middle tier service is shown as follows:

```

| TCP/IP | encryption | http | web response |
| dispatcher response | proxy-specific response |

```

where | separates a logical protocol level, and protocols nested from left to right.

The foregoing merely illustrates the principles of the present invention. Those skilled in the art will be able to devise various modifications, which although not explicitly described or shown herein, embody the principles of the invention and are thus within its spirit and scope.

What is claimed is:

1. A Web/Internet based reporting system for providing timely delivery of a customer's priced telecommunications call detail data to a client workstation running a web browser application, said system comprising:

a data warehousing infrastructure comprising:

process for generating a current customer list on a daily basis comprising customers entitled to receive daily telecommunications call detail data;

device for receiving customer's raw telecommunications call detail data records from one or more telecommunications network switch mechanisms, and extracting certain call detail records for pre-determined customers;

harvest device for receiving said extracted call detail data records and replacing a call detail data item therein with a corresponding dimension key found in an associated dimension build table for that call detail item; and,

device for generating an output fact table comprising customer records having unique key structures for enabling consolidated storage of specific customer call detail data;

at least one secure server for managing client sessions over the Internet, the secure server supporting secure communication of customer request messages between the browser application client and the secure server; and,

device for receiving said customer requests from said secure server and generating corresponding database queries implementing said dimension keys for application against said output fact table to obtain a specific call customer's call detail data, said call detail data being transmitted back to said client web browser via said secure server;

51

whereby expedient and updated web/Internet-based access to said customer's daily call detail data is assured.

2. The system as claimed in claim 1, wherein said dimension build tables are continuously generated and updated with key structures for each extracted telecommunications call detail data record having a call detail item in which no associating match is found.

3. The system as claimed in claim 1, wherein said harvest device applies look-ups to each of said one or more dimension build tables having said unique dimension keys, said harvest device further replacing a call detail data item therein with a corresponding key structure when a match is found.

4. The system as claimed in claim 2, wherein a dimension table includes a dimension key associated with a particular billing/customer, said system further comprising process for updating said billing/customer dimension table prior to inputting a received modified call detail record to said harvest device.

5. The system as claimed in claim 3, wherein said updating of said billing/customer dimension includes process for comparing said current customer list with a customer list from a prior point in time, wherein new customers in said current daily list is added to said billing/customer dimension build table.

6. The system as claimed in claim 3, further including a device for organizing said fact table and dimension tables in one or more operational data storage devices, said operational data storage devices comprising a database server for facilitating expedient retrieval of customer's daily call detail data upon received customer requests.

7. The system as claimed in claim 3, further including a translation process for converting call detail data in said fact table from a first character format to a second character format, and transferring said fact table data to said operational data storage devices via ftp protocol.

8. The system as claimed in claim 1, wherein said output fact table and said dimension build tables are organized according to a star schema structure.

9. The system as claimed in claim 2, wherein said dimension build tables include dimension keys associated with a particular calling area.

10. The system as claimed in claim 5, further including dispatch server for communicating with said secure server through a firewall over a second socket connection, the first and second secure sockets forming a secure communications link, said dispatch server enabling forwarding of a report request message and an associated report response message comprising requested call detail data back to the client browser over the secure communications link.

11. The system as claimed in claim 9, further including a report manager server for maintaining an inventory of priced call detail reporting items associated with a customer and managing the reporting of daily call-detail data information in accordance with a customer request message, the report manager initiating access to said customer daily call detail data from said operational data stores.

12. The system as claimed in claim 5, wherein said data warehousing infrastructure further includes process for totaling each customer's daily call detail data for a predetermined period of time.

13. The system as claimed in claim 12, further including process for verifying data harvesting and database load totals.

14. The system as claimed in claim 12, wherein said data warehousing infrastructure is further capable of providing

52

output fact table associated with customer's monthly call detail data that has been repriced, said system further including process for reconciling a customer's daily call detail data totals in said operational data storage device with that customer's repriced monthly call detail data.

15. The system as claimed in claim 14, wherein said reconciling process includes process for fragmenting blocks of call detail data included in said operational data storage devices with repriced monthly call detail data.

16. A method for providing timely delivery of a customer's priced telecommunications call detail data to a client workstation running a web browser application comprising the steps of:

generating a current customer list on a daily basis comprising customers entitled to receive daily telecommunications call detail data;

receiving customer's daily raw telecommunications call detail data records from one or more telecommunications network switch mechanisms, and extracting those call detail data records for only those customers entitled to receive said call detail data;

receiving said extracted call detail data records and replacing one or more call detail data items therein with a corresponding dimension key found in a dimension build table associated with said call detail item; and,

generating an output fact table comprising customer records having said unique key structures for enabling consolidated storage of specific customer call detail data;

implementing a secure server to manage client sessions over the Internet, the secure server supporting secure communication of customer request messages between the browser application client and the secure server; and,

receiving said customer requests from said secure server and generating corresponding database queries implementing said dimension keys for application against said output fact table to obtain a specific customer's call detail data, said call detail data being transmitted back to said client web browser via said secure server, whereby expedient and updated web/Internet-based access to said customer's daily call detail data is assured.

17. The method as claimed in claim 16, wherein said step of replacing a call detail data item therein with a corresponding dimension key includes implementing a dimension table look-up of the call detail item for a key structure corresponding to said call detail item and replacing said key in said record when a match is found.

18. The method as claimed in claim 17, wherein said step of replacing said call detail item with said corresponding dimension key further includes the step of generating a new key to replace said call detail item and providing said new key to said dimension build table when no match is found.

19. The method as claimed in claim 18, wherein a dimension table includes a dimension key associated with a particular billing/customer, said method further updating said billing/customer dimension table prior to inputting a received modified call detail record to a harvest device.

20. The method as claimed in claim 19, wherein said step of updating said billing/customer dimension includes comparing said current daily customer list with a customer list from an immediate prior day, wherein new customers in said current daily list is added to said billing/customer dimension build table.

21. The method as claimed in claim 20, wherein said output fact table is capable of being loaded into storage/

53

retrieval devices, said method further including the step of organizing said fact table and dimension tables in one or more operational data storage devices, said operational data storage devices comprising a database server for retrieving customer's daily call detail data upon received customer requests.

22. The method as claimed in claim 21, further including the step of converting call detail data in said fact table from a first character format to a second character format, and transferring said fact table data to said operational data storage devices via ftp protocol.

23. The method as claimed in claim 21, wherein said fact table and dimension tables are organized according to a star schema structure.

24. The method as claimed in claim 23, wherein a dimension table includes dimension keys associated with a particular calling location.

25. The method as claimed in claim 24, further including the step of forwarding a report request message and an associated report response message containing requested call detail data back to the client browser over a secure communications link.

26. The method as claimed in claim 25, further including maintaining an inventory of priced call detail reporting items associated with a customer and managing the reporting of daily call-detail data information in accordance with a customer request message, and initiating access to said customer daily call detail data stored in said operational data stores.

27. The method as claimed in claim 26, further including the step of totaling one or more call detail data item values for a predetermined period of time.

28. The method as claimed in claim 26, further providing an output fact table associated with customer's monthly call detail data that has been repiced, said method further including reconciling a customer's daily call detail item value totals in said operational data storage device with that customer's repiced monthly call detail data.

29. The method as claimed in claim 28, wherein said reconciling step includes fragmenting blocks of call detail data included in said operational data storage devices with repiced monthly call detail data.

30. The method as claimed in claim 26, further including interfacing with a report manager server for retrieving the customer-specific data from the operational data stores in accordance with a customer identity and report name.

31. The method as claimed in claim 26, further including generating a report utilizing the retrieved data and the metadata description of call detail reporting items.

54

32. A Web/Internet based reporting system for providing timely delivery of a customer's priced telecommunications call detail data to a client workstation running a web browser application, said system comprising:

a data warehousing infrastructure comprising:

a traffic component to receive raw telecommunications call detail records from one or more telecommunications switches and to sort the call detail records into billing detail records;

a billing system to perform pricing in connection with one or more billing detail records;

an extraction component to extract selected customer billing detail records;

a harvesting component to transform the selected billing detail records based on a set of business rules into transformed billing detail records;

a data store component to store the transformed billing detail records into one or more data marts;

at least one secure server for managing client sessions over the Internet, the secure server supporting secure communication of customer requests between the browser application client and the secure server; and, a device for receiving the customer requests from the secure server and generating corresponding database queries to obtain a specific customer's call detail data, the call detail data being transmitted back to said client web browser via the secure server.

33. The system of claim 32 wherein the traffic component comprises a module to convert the raw telecommunications call detail records into a format that is readable by a mainframe.

34. The system of claim 32 wherein the traffic component comprises a module to determine if records are billable and drops unbillable records.

35. The system of claim 32 wherein the extraction component comprises a file transfer protocol pull of a list of selected customers.

36. The system of claim 32 wherein the set of business rules for transforming the selected customer billing detail records comprises a star schema data model which incorporates a central fact table and a plurality of referenced dimension tables.

37. The system of claim 32 wherein the data warehousing infrastructure comprises a component to provide incremental, daily updates to the transformed billing detail records stored in the one or more data marts.

* * * * *



US006567122B1

(12) United States Patent
Anderson et al.**(10) Patent No.: US 6,567,122 B1**
(45) Date of Patent: May 20, 2003**(54) METHOD AND SYSTEM FOR HOSTING AN INTERNET WEB SITE ON A DIGITAL CAMERA**2001/0010543 A1 * 8/2001 Ward et al. 348/207.99
2001/0014910 A1 * 8/2001 Bobo, II 709/217
2001/0050711 A1 * 12/2001 Kanbe et al. 348/220**(75) Inventors:** Eric C. Anderson, San Jose, CA (US);
Michael A. Ramirez, Palo Alto, CA (US);
Stephen G. Sullivan, Mountain View, CA (US)**FOREIGN PATENT DOCUMENTS**
EP 0 821 522 2/1998 H04N/5/232
EP 0635011 8/1998
WO 96/02106 1/1996 H04N/5/232
WO 97/38510 10/1997 H04L/12/58**(73) Assignee:** IPAC Acquisition Subsidiary I,
Peterborough, NH (US)**OTHER PUBLICATIONS****(*) Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

PATNEWS: What isn't obvious in the patent world? E-Mail correspondence.

* cited by examiner

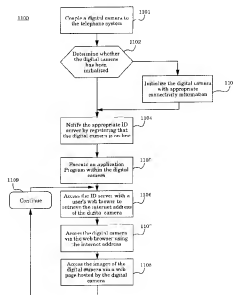
Primary Examiner—Wendy R. Garber

Assistant Examiner—Lin Ye

(74) Attorney, Agent, or Firm—Sawyer Law Group LLP

(21) Appl. No.: 09/044,644**(22) Filed: Mar. 18, 1998****(51) Int. Cl.⁷** H04N 5/232; H04N 9/04;
H04N 7/14; G06F 15/16**(52) U.S. Cl.** 348/211.3; 348/207.99;
348/14.01; 709/201; 709/217**(58) Field of Search** 348/552, 207,
348/14, 14.03, 207.99, 207.1, 207.11, 211.3,
14.01; 709/201, 217, 219**(56) References Cited****U.S. PATENT DOCUMENTS**5,633,678 A * 5/1997 Paulski et al. 348/232
5,649,186 A * 7/1997 Ferguson 707/10
5,999,207 A * 12/1999 Rodriguez et al. 348/14
6,012,088 A * 1/2000 Li et al. 709/219
6,035,323 A * 3/2000 Narayan et al. 709/201
6,163,335 A * 12/2000 Barraclough 348/15
6,205,485 B1 * 3/2001 Kikinis 709/231
6,246,430 B1 * 6/2001 Peters et al. 348/14
6,353,648 B1 * 3/2002 Morris 709/203**(57) ABSTRACT**

The present invention comprises a method and system for implementing internet access to images stored in a digital image capture unit including an imaging device and a display. The image capture unit (e.g., a digital camera) is used to capture images and store them within its internal memory. The image capture unit accesses a ID server via the internet and registers its identity and internet address with the web server. A user subsequently accesses the image capture unit by entering the identity of the image capture unit into his web browser. The web browser, using standard internet protocols, then queries the ID server with the identity of the image capture unit and retrieves the internet address. The internet address is subsequently used to access a web page hosted by the image capture unit and display the web page to the user. The web page provides access to the stored images within the image capture unit.

22 Claims, 11 Drawing Sheets

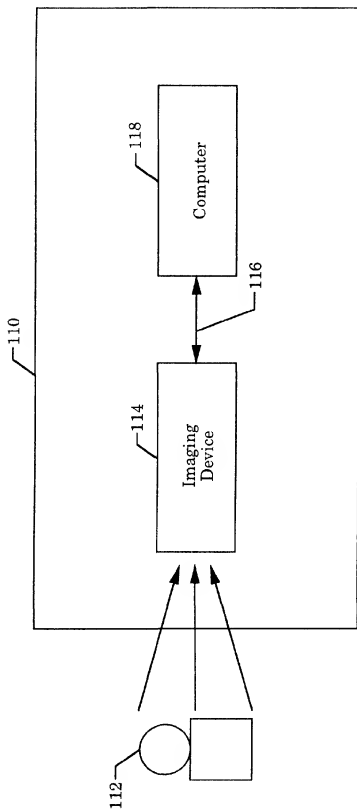


FIG. 1

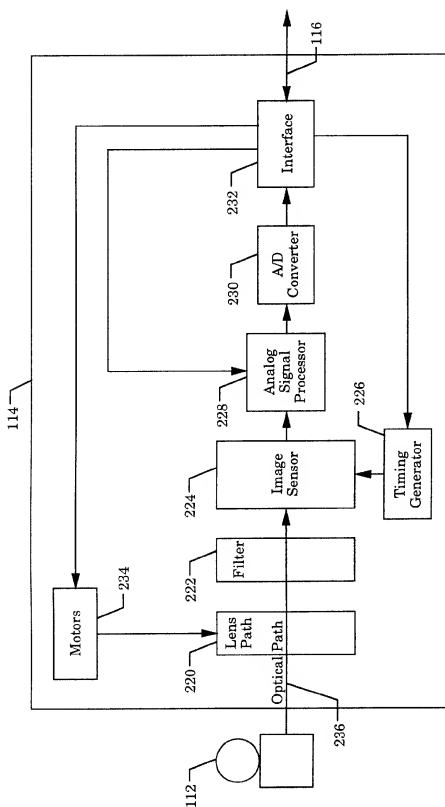
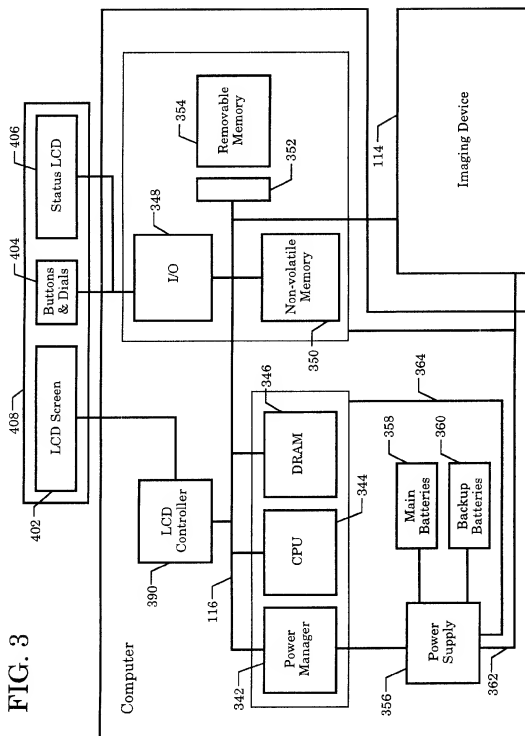


FIG. 2

FIG. 3



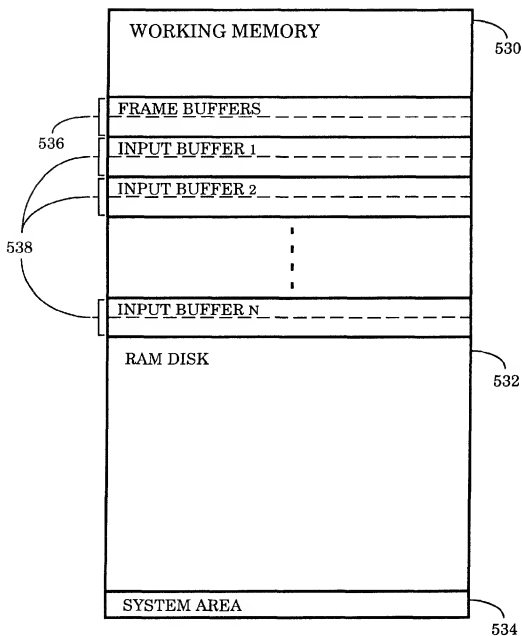


FIG. 4

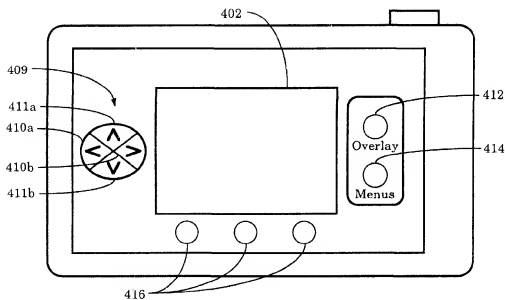


FIG. 5A

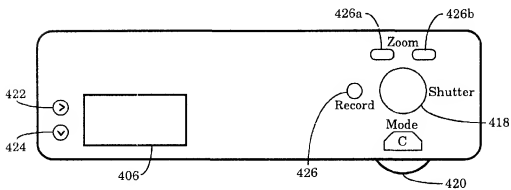


FIG. 5B

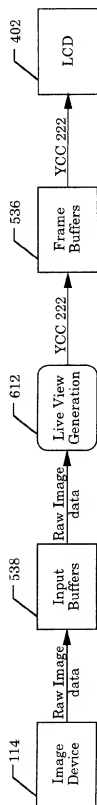


FIG. 6

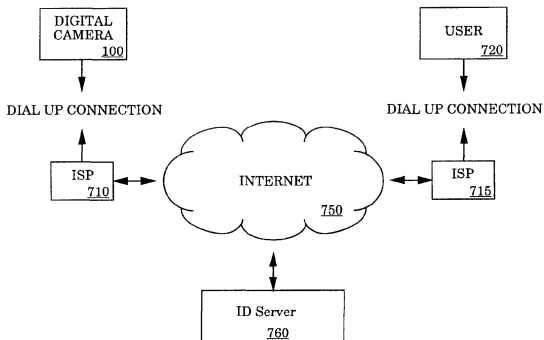
700

FIG. 7

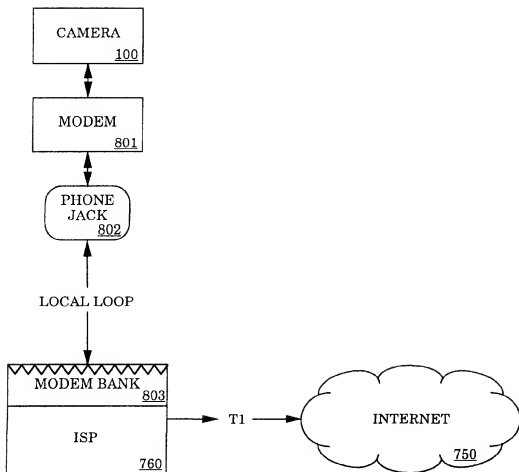


FIG. 8

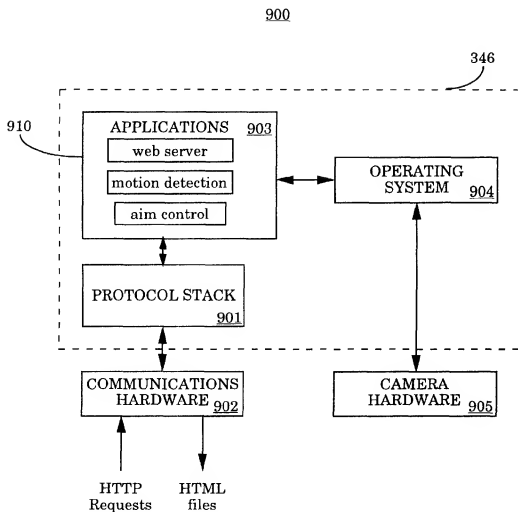


FIG. 9

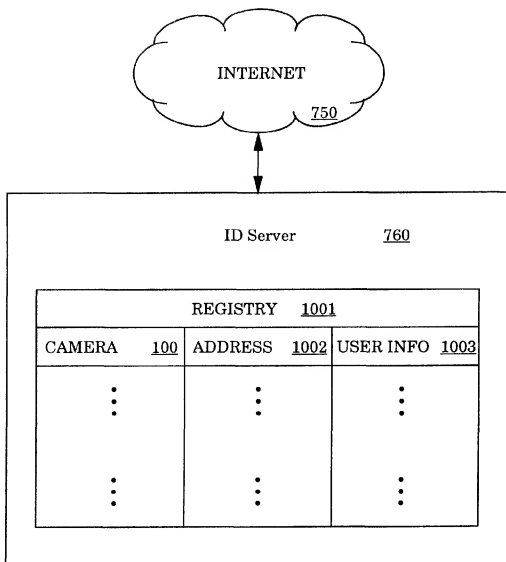


FIG. 10

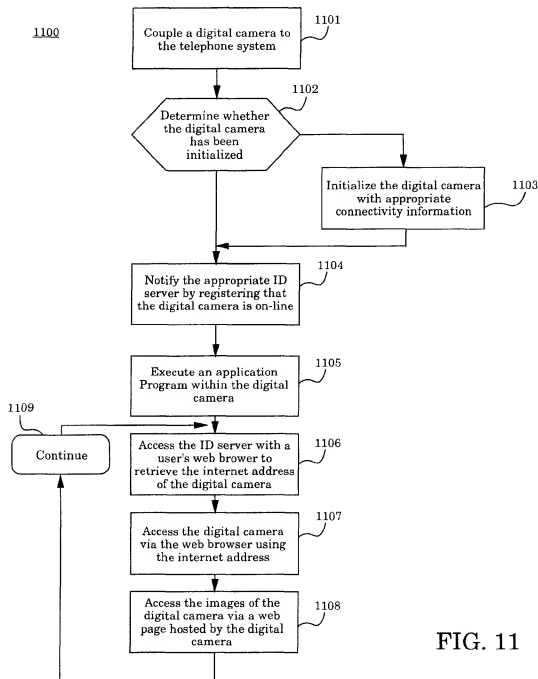


FIG. 11

METHOD AND SYSTEM FOR HOSTING AN INTERNET WEB SITE ON A DIGITAL CAMERA

FIELD OF THE INVENTION

The field of the present invention pertains to digital image capture devices. More particularly, the present invention relates to a method and system for using the electronic systems within a digital camera with the internet.

BACKGROUND OF THE INVENTION

Modern digital cameras for taking pictures of scenes and the like typically include an imaging device which is controlled by a computer running a software program. When an image is captured, the imaging device is exposed to light and generates raw image data representing the image. The raw image data is typically stored in a single image buffer where it is then processed and compressed by the processor. Many types of compression schemes are used to compress the image data, with the joint photographic expert group (JPEG) standard being the most popular. After the processor processes and compresses the raw image data into JPEG image files, the processor stores the JPEG image files into an internal memory or on an external memory card.

Some digital cameras are also equipped with a liquid-crystal display (LCD) or other type of display screen on the back of the camera. Through the use of the LCD, the processor can cause the digital camera to operate in one of two modes, play and record, although some cameras only have a record mode. In play mode, the LCD is used as a playback screen for allowing the user to review previously captured images either individually or in arrays of four, nine, or sixteen images. In record mode, the LCD is used as a viewfinder in which the user may view an object or scene before taking a picture.

Besides the LCD, digital camera user interfaces also include a number of buttons or switches for setting the camera into one of the two modes and for navigating between images in play mode. For example, most digital cameras include two buttons labeled "-" and "+" that enable a user to navigate or scroll through captured images. For example, if the user is reviewing images individually, meaning that single images are displayed full-sized in the LCD, pressing one of navigation buttons causes the currently displayed image to be replaced by the next image.

It should be noted that a digital camera has no "film", and as such, there is no incremental cost of taking and storing pictures. Within the confines of memory, the cost taking and storing each additional picture is insignificant. For a given memory size, it is possible to take an unlimited number of pictures, wherein the most recent picture replaces the earliest picture, for virtually zero incremental cost. Accordingly, this advantage is best realized when the camera is used as much as possible, taking pictures of practically anything of interest.

One way to best utilize these unique attributes is to make the digital camera and its internally stored images remotely accessible. If the pictures are remotely accessible, the camera could be set to continuously take pictures of scenes/items of interest. Ideally, a user would be able to access those pictures at any time. The user would be able to use a widely available communications medium to access the camera from virtually an unlimited number of locations.

The emergence of the internet as a distributed, widely accessible communications medium provides a convenient

avenue for implementing remote accessibility. Providing remote accessibility via the internet leverages the fact that the internet is becoming increasingly familiar to increasing numbers of people. Many users have become accustomed to retrieving information from remotely located systems via the internet. There are many and varied applications which presently use the internet to provide remote access or remote connectivity. Internet telephony is one such application, such as, for example, Microsoft's NetMeeting and Netscape's CoolTalk.

NetMeeting and CoolTalk are both real-time desktop audio conferencing and data collaboration software applications specifically designed to use the internet as their communications medium. Both software applications allow a "local" user to place a "call" to a "remote" user located anywhere in the world. With both NetMeeting and CoolTalk, the software application is hosted on a personal computer at the user's location and on a personal computer at the remote user's location. Both NetMeeting and CoolTalk require a SLIP (Serial Line Internet Protocol) or PPP (Point-to-Point Protocol) account where internet access is via a dial-up modem, where the user, as is typical, accesses the internet through their respective ISP (internet service provider). Both NetMeeting and CoolTalk require personal computers for the necessary resources for running the applications (e.g., processing power, memory, communications hardware, etc.). In addition, both NetMeeting and CoolTalk require the one user to input an IP (Internet Protocol) address for the other user to input to establish communication between them. To place a call, for example, the local user enters the IP address of the remote user in an appropriate field of the software application and subsequently initiates the call (e.g., by clicking a graphic icon on the personal computer's display), which in turn, establishes communication between the users.

To facilitate the process of obtaining appropriate internet addresses, CoolTalk, for example, allows on-line users to list their respective IP addresses with a proprietary central CoolTalk server. This allows a user to obtain a list of currently on-line users to whom communication can be established. Upon locating the desired remote user in the web server maintained internet address list, the local user places the call.

In this manner, the proprietary central CoolTalk server maintains a user viewable, user updated, "address book" in which users list their respective internet addresses and in which they search for the internet addresses of others with whom they wish to communicate. However, as described above, both NetMeeting and CoolTalk require active user input, in that each require the user to input his current internet address and in that each require the user to search the address book for the internet address of the individual to be contacted. This can be quite problematic in the case where users obtain access to the internet via dial-up connections, and hence, have different internet addresses each time their respective dial-up connections are established.

In a manner similar to internet telephony, internet desktop video conferencing is another application which uses the internet as its communications medium. One such application, for example, is CU-SeeMe, by White Pine. CU-SeeMe provides real time video conferencing between two or more users. As with NetMeeting and CoolTalk, CU-SeeMe is a software application which runs on both the local user's personal computer and the remote user's personal computer. The personal computers provide the resources for running the application. As with NetMeeting

and CoolTalk, CU-SecMe requires the local user to enter the IP address of the remote user. CU-SecMe also facilitates this process by allowing on-line users to list their respective IP addresses with a proprietary central server such that the addresses can be easily indexed and searched.

Another example of remote access via the internet is status queries of remote devices using the internet as the communications medium. A typical prior art application involves interfacing a remote device with a computer system, and providing access to the computer system via the internet. For example, a vending machine can be remotely accessed to determine its status (e.g., the number of sales made, whether the machine needs refills, whether the machine needs maintenance, etc.). The machine is appropriately equipped with sensors, switches, and the like, which are in turn, interfaced to a computer system using a software driver. The computer system is coupled to the internet and interfaces with the machine through the driver, making the relevant information available over the internet using web server software. Hence, any interested user (e.g., the vending machine service company) is able to remotely ascertain the status of the machine via the internet.

The problem with the above described prior art applications is that access to the internet and communication thereon requires a separate host computer system (e.g., a personal computer). Each of the above described applications (CoolTalk, NetMeeting, and the vending machine examples) require a computer system on both sides of the internet connection. The two computer systems provide the computational resources to host the respective software application, the internet access software, and any necessary device drivers. Because of this, among other reasons, the above applications are not easily transferred to the realm of easy-to-use, intuitive, consumer electronic type devices such as digital cameras. The separate computer systems are expensive.

Another problem is the fact that the above applications require the user to know the internet address of the person (or device, in the vending machine example) being contacted. The internet telephony applications (e.g., CoolTalk) often employ a user viewable, user updated, address book to facilitate the process of locating and obtaining the correct internet address; however, they require active user input. This is difficult in the case where users obtain access to the internet via dial-up connections, and thus, have changing internet addresses.

In addition, both CoolTalk and NetMeeting operate on top of the computer's operating system, which is notoriously difficult and obtuse to novice users.

Thus, what is required is an inexpensive method implementing remote access via the internet for digital cameras. If internet remote accessibility is relatively inexpensive, a large installed base of remotely accessible digital cameras will rapidly develop. This will give rise to many different applications and enhancements being developed, which in turn, will lead to even greater demand for, and use of, remotely accessible digital cameras. What is further required is an intuitive, easy to use interface for presenting the digital camera's functionality and capabilities to users. Additionally, what is required is an efficient, user transparent, process of obtaining the internet address of a digital camera, where the camera accesses the internet via a dialup connection, and thus, has a changing internet address. The present invention provides a novel solution to the above requirements.

SUMMARY OF THE INVENTION

The present invention provides a method for making a digital camera and its internally stored images remotely

accessible. The present invention enables the digital camera to be set to continuously take pictures of scenes/items of interest and allow a user to access those pictures at any time. The present invention implements remote accessibility via the internet. This allows the both the user and the digital camera to communicate from virtually an unlimited number of locations. Hence, both the user and the camera are portable, requiring only an internet connection at any location to implement remote access. The present invention also provides an efficient, user transparent, process of obtaining the internet address of a digital camera, where the camera accesses the internet via a dial-up connection, and thus, has a changing internet address.

A digital camera in accordance with the present invention does not require a separate, external computer system (e.g., a personal computer) for internet connectivity, thus providing an inexpensive method for making remotely accessible digital cameras widely available.

In addition, a digital camera in accordance with the present invention is accessed via the widely used, very familiar web browser. By functioning with typical, widely used web browsers, the present invention provides a simple, intuitive, and familiar interface for accessing the digital camera's functionality. Accordingly, the digital camera's controls and functions are intuitively easy to utilize, without requiring a extensive learning period for new users. For example, a consumer purchasing a remotely accessible camera is typically able to easily and immediately use the remote accessibility functions with minimal set-up.

In one embodiment, the present invention comprises a method and system for implementing internet access to images stored in a digital camera including an imaging device and a display. The digital camera (e.g., or similar image capture unit) is used to capture images and store them within its internal memory. The digital camera accesses a ID server via the internet and registers its identity and internet address with the web server.

A user wishing to view the image (e.g., the camera's owner or any other user) subsequently enters the identity of the digital camera into his web browser (e.g., the camera's URL). Using standard internet protocols, the ID server is queried with the URL of the digital camera and returns the digital camera's current internet address. The user's web browser then accesses the digital camera using the camera's current internet address returned from the ID server, and views web pages hosted by the camera. This process of retrieving the current internet address of the digital camera from the ID server occurs transparently with respect to the user. The web page provides access to the stored images within the digital camera. By functioning with typical, widely used web browsers, the digital camera of the present invention provides a simple, intuitive, and familiar interface for accessing the digital camera's functionality. And by implementing remote accessibility via the internet, the present invention allows access to the digital camera from virtually an unlimited number of locations.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not by way of limitation, in the figures of the accompanying drawings in which like reference numerals refer to similar elements and in which:

FIG. 1 shows a block diagram of a digital camera for use in accordance with the present invention.

FIG. 2 shows a block diagram of an imaging device in accordance with one preferred embodiment of the present invention.

5

FIG. 3 shows a block diagram of a computer in accordance with one preferred embodiment of the present invention.

FIG. 4 shows a memory map of a DRAM in accordance with one embodiment of the present invention.

FIG. 5A shows a top view diagram depicting the preferred hardware components of the camera from FIG. 1.

FIG. 5B shows a back view diagram depicting the preferred hardware components of the camera from FIG. 1.

FIG. 6 shows a block diagram of a live view generation process in accordance with one embodiment of the present invention.

FIG. 7 shows a block diagram of a remote access system in accordance with one embodiment of the present invention.

FIG. 8 shows a block diagram of the digital camera from FIG. 7 coupled to the internet via an internet service provider.

FIG. 9 shows a diagram of the connectivity and application software of a digital camera in accordance with one embodiment of the present invention.

FIG. 10 shows a more detailed diagram of the domain name server from FIG. 7.

FIG. 11 shows a flow chart of a process in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In the following detailed description of the present invention, numerous specific details are set forth in order to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Although the present invention will be described in the context of a digital camera, various modifications to the preferred embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. That is, any image capture device which displays images, icons and/or other items, could incorporate the features described herein and that device would be within the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

The present invention provides a method for making a digital camera and its internally stored images remotely accessible by hosting an internet web site on the digital camera. The present invention enables the digital camera to be set to continuously take pictures of scenes/items of interest and allow a user to access those pictures at any time. The present invention implements remote accessibility via the internet, thus allowing the user to access the digital camera from virtually an unlimited number of locations.

A digital camera in accordance with the present invention does not require a separate, external computer system (e.g., a personal computer) for internet connectivity, thus providing an inexpensive method for making remotely accessible digital cameras widely available. In addition, a digital camera in accordance with the present invention is accessed via the widely used, very familiar web browser. By functioning with typical, widely used web browsers, the present invention provides a simple, intuitive, and familiar interface for accessing the digital camera's functionality. In so doing, the controls and functions of the digital camera are intuitively easy to utilize, and do not require an extensive learning

6

period for new users. These and other benefits of the present invention are described in greater detail below.

Referring now to FIG. 1, a block diagram of a digital camera 110 is shown for use in accordance with the present invention. Camera 110 preferably comprises an imaging device 114, a system bus 116 and a computer 118. Imaging device 114 is optically coupled to an object 112 and electrically coupled via system bus 116 to computer 118. Once a photographer has focused imaging device 114 on object 112 and, using a capture button or some other means, instructed camera 110 to capture an image of object 112, computer 118 commands imaging device 114 via system bus 116 to capture raw image data representing object 112. The captured raw image data is transferred over system bus 116 to computer 118 which performs various image processing functions on the image data before storing it in its internal memory. System bus 116 also passes various status and control signals between imaging device 114 and computer 118.

Referring now to FIG. 2, a block diagram of one preferred embodiment of imaging device 114 is shown. Imaging device 114 typically comprises a lens 220 having an iris, a filter 222, an image sensor 224, a timing generator 226, an analog signal processor (ASP) 228, an analog-to-digital (A/D) converter 230, an interface 232, and one or more motors 234.

In operation, imaging device 114 captures an image of object 112 via reflected light impacting image sensor 224 along optical path 236. Image sensor 224, which is typically a charged coupled device (CCD), responsively generates a set of raw image data in CCD format representing the captured image 112. The raw image data is then routed through ASP 228, A/D converter 230 and interface 232. Interface 232 has outputs for controlling ASP 228, motors 234 and timing generator 226. From interface 232, the raw image data passes over system bus 116 to computer 118.

Referring now to FIG. 3, a block diagram of one preferred embodiment for computer 118 is shown. System bus 116 provides connection paths between imaging device 114, an optional power manager 342, central processing unit (CPU) 344, dynamic random-access memory (DRAM) 346, input/output interface (I/O) 348, non-volatile memory 350, and buffers/connector 352. Removable memory 354 connects to system bus 116 via buffers/connector 352. Alternately, camera 110 may be implemented without removable memory 354 or buffers/connector 352.

Power manager 342 communicates via line 366 with power supply 356 and coordinates power management operations for camera 110. CPU 344 typically includes a conventional processor device for controlling the operation of camera 110. In the preferred embodiment, CPU 344 is capable of concurrently running multiple software routines to control the various processes of camera 110 within a multithreaded environment. DRAM 346 is a contiguous block of dynamic memory which may be selectively allocated to various storage functions. LCD controller 390 accesses DRAM 346 and transfers processed image data to LCD screen 402 for display.

I/O 348 is an interface device allowing communications to and from computer. For example, I/O 348 permits an external host computer (not shown) to connect to and communicate with computer 118. I/O 348 also interfaces with a plurality of buttons and/or dials 404, and an optional status LCD 406, which in addition to the LCD screen 402, are the hardware elements of the camera's user interface 408.

7

Non-volatile memory 350, which may typically comprise a conventional read-only memory or flash memory, stores a set of computer-readable program instructions to control the operation of camera 110. Removable memory 354 serves as an additional image data storage area and is preferably a non-volatile device, readily removable and replaceable by a camera 110 user via buffers/connector 352. Thus, a user who possesses several removable memories 354 may replace a full removable memory 354 with an empty removable memory 354 to effectively expand the picture-taking capacity of camera 110. In the preferred embodiment of the present invention, removable memory 354 is typically implemented using a flash disk. Power supply 356 supplies operating power to the various components of camera 110. In the preferred embodiment, power supply 356 provides operating power to a main power bus 362 and also to a secondary power bus 364. The main power bus 362 provides power to imaging device 114, I/O 348, non-volatile memory 350 and removable memory 354. The secondary power bus 364 provides power to power manager 342, CPU 344 and DRAM 346.

Power supply 356 is connected to main batteries 358 and also to backup batteries 360. In the preferred embodiment, a camera 110 user may also connect power supply 356 to an external power source. During normal operation of power supply 356, the main batteries 358 provide operating power to power supply 356 which then provides the operating power to camera 110 via both main power bus 362 and secondary power bus 364. During a power failure mode in which the main batteries 358 have failed (when their output voltage has fallen below a minimum operational voltage level) the backup batteries 360 provide operating power to power supply 356 which then provides the operating power only to the secondary power bus 364 of camera 110.

Referring now to FIG. 4, one embodiment of dynamic random-access-memory (DRAM) 346 is shown. In the preferred embodiment, DRAM 346 includes RAM disk 532, a system area 534, and working memory 530.

RAM disk 532 is a memory area used for storing raw and compressed image data and typically is organized in a "sectored" format similar to that of conventional hard disk drives. In the preferred embodiment, RAM disk 532 uses a well-known and standardized file system to permit external host computer systems, via I/O 348, to readily recognize and access the data stored on RAM disk 532. System area 534 typically stores data regarding system errors (for example, why a system shutdown occurred) for use by CPU 344 upon a restart of computer 118.

Working memory 530 includes various stacks, data structures and variables used by CPU 344 while executing the software routines used within computer 118. Working memory 530 also includes several input buffers 538 for temporarily storing sets of raw image data received from imaging device 114, and a frame buffer 536 for storing data for display on the LCD screen 402. In a preferred embodiment, each input buffer 538 and the frame buffer 536 are split into two separate buffers (shown by the dashed lines) to improve the display speed of the digital camera and to prevent the tearing of the image in the display 402.

FIGS. 5A and 5B are diagrams depicting the preferred hardware components of the camera's 110 user interface 408. FIG. 5A is back view of the camera 110 showing the LCD screen 402, a four-way navigation control button 409, an overlay button 412, a menu button 414, and a set of programmable soft keys 416. FIG. 5B is a top view of the camera 110 showing a shutter button 418, and a mode dial

8

420. The camera may optionally include status LCD 406, status LCD scroll and select buttons 422 and 424, a sound record button 426, and zoom-in, zoom-out buttons 426a and 426b.

In the present embodiment, the digital camera is provided with several different operating modes for supporting various camera functions. In capture mode, the camera 100 supports the actions of preparing to capture an image, and capturing an image through the use of either the LCD screen 402 alone or the status LCD 406 with the aid of an optional optical viewfinder (not shown). In review mode, the camera 100 supports the actions of reviewing camera contents, editing and sorting images, and printing and transferring images. In play mode, the camera 100 allows the user to view screen-sized images in the LCD screen 402 in the orientation that the image was captured. Play mode also allows the user to hear recorded sound associated to a displayed image, and to play back sequential groupings of images, which may comprise time lapse, slide show, and burst image images. The user preferably switches between the capture, review, and play modes, using the mode dial 420. When the camera is placed into a particular mode, that mode's default screen appears in the LCD screen 402 in which a set of mode-specific items, such as images, icons, and text, are displayed. Although the digital camera includes multiple operating modes, the mode relevant to this description is capture (record) mode.

Referring now to FIG. 6, in a preferred embodiment, the processing is performed by a live view generation process 612, which is stored in non-volatile memory 350 and executed on CPU 344. However, the image processing can also be implemented using hardware. During the execution of the live view generation process 612, the CPU 344 takes the raw image data from the input buffers 538 and performs image processing and color space conversion. Image processing steps can include, for example, gamma correction, white balance, and color correction. The conversion process performs gamma correction and converts the raw CCD data into either a RGB or YCC color format which is compatible with the LCD screen 402. (RGB is an abbreviation for Red, Green, Blue, and YCC is an abbreviation for Luminance, Chrominance-red and Chrominance-blue). After converting the data to YCC, the YCC image data is stored in the frame buffer 536. The contents of the frame buffer 536 are then displayed onto the LCD screen 402. Although FIG. 6 shows the YCC data being displayed on LCD 402, it should be appreciated that the present invention is not limited to functioning only with LCD equipped digital cameras.

Referring now to FIG. 7, a block diagram of a remote access system 700 in accordance with one embodiment of the present invention is shown. System 700 includes camera 100, internet service provider (ISP) 710, internet service provider 715, and user 720. ISP 710 and ISP 715 are both directly coupled to the internet 750. System 700 also includes a ID server 760. In the present embodiment, ID server 760 includes the functionality of a domain name server.

ID server 760 functions in part by facilitating the process of locating appropriate internet addresses. As is well known in the art, web sites are found and web pages are accessed on the internet 750 via their internet addresses. URIs are referred to corresponding internet addresses. The URIs are the universal naming scheme for identifying and locating all web resources. URIs, or internet addresses, fully describe where a particular resource (e.g., a web page) resides and how to access it. Using well known internet techniques (e.g., hypertext transfer protocol), resources which exist in "internet space" are located and accessed via their internet addresses.

There is a problem, however, in that each time camera 100 dials up and connects to internet 750 via ISP 710, it typically is assigned a different internet address.

In the case of dial-up internet access, the actual internet address is not assigned by ISP 710 until the device actually establishes an internet connection. The internet address typically changes each time camera 100 establishes an internet connection. Thus, for any particular session, when user 720 attempts to access camera 100 via the internet 750, wherein camera 100 is connected to the internet 750 (e.g., via a dial-up connection) user 720 will not know the correct internet address. ID server 760, in accordance with the present invention, overcomes this unknown address problem and allows access to the digital camera (e.g., digital camera 100), which inexpensively hosts a web site, as described further below. ID server 760 is further described in the discussion of FIG. 10 below.

With reference still to FIG. 7, process 700 of the present invention provides a method which implements remote access to camera 100 and its internally stored images. In the present embodiment, camera 100 is coupled to the internet 750 via a dial up connection to ISP 710. The dial up connection is via a POTS (plain old telephone system) telephone line. Digital camera 100 accesses ISP 710 using a modem, coupling to one of a bank of modems maintained on the premises of ISP 710. ISP 710 is in turn coupled directly to the internet 750 via an all-digital connection (e.g., T1 line).

Similarly, user 720 is coupled to ISP 715 via a POTS dial up connection and is likewise coupled to the internet 750 via one of a bank of modems maintained on the premises of ISP 715. As with ISP 710, ISP 715 is coupled directly to the internet via an all-digital connection. User 720 accesses the internet 750 using a web browser (not shown) running on any one of a variety of devices (e.g., personal computer, wireless PCS phone, network computer, television set top box, etc.).

Camera 100 accesses ID server 760 via the internet 750 and registers its identity and internet address. ID server 760 maintains an internal database of "on-line" devices and their associated internet addresses. User 720, or any other user wishing to access the camera (e.g., the camera owner's friends or relatives) subsequently enters the identity of camera 100 into his web browser (e.g., camera 100's URL). Using standard internet protocols, ID server 760 is queried with the URL of camera 100 and returns the camera 100's current internet address. The user 720's web browser then accesses camera 100 using the current internet address returned from ID server 760.

After the current internet address of camera 100 is returned from ID server 760, user 720's web browser accesses camera 100 to retrieve a web page. The web browser embeds the internet address inside the HTTP (Hypertext Transfer Protocol) request and sends the request, along with some status information, to a web server application hosted by camera 100 (e.g., server application 910 shown in FIG. 9). Web server application 910 receives the HTTP request and establishes a socket connection between user 720's web browser and web server application 910. Web server application 910 subsequently fetches the requested HTML (Hypertext Mark-up Language) file and sends it back to the web browser and closes the socket connection. The web browser then interprets the HTML commands and displays the resulting web page. The process of accessing an HTML file from a web server is commonly referred to as accessing a web page. Similarly, the process of sending HTML files

from a web server to a web browser is commonly referred to as sending a web page, and hosting the web server which sends the web page is often referred to as hosting the web page.

This process of retrieving the current internet address of camera 100 from ID server 760 occurs transparently with respect to user 720. In a typical case, for example, user 720 types the URL for camera 100 into his web browser and hits enter. In accordance with the present invention, the next web page the user views is the web page returned from camera 100. Beyond entering the URL for camera 100, no further action from the user is required in order to access the web pages hosted by camera 100.

Web server application 910 (FIG. 9) hosted by camera 100 provides access to the stored images via the web pages. For example, requested images are embedded within the web pages which are sent to user 720's web browser. And user 720's web browser requests images or issues commands to camera 100, by embedding them within the status information included within the HTTP requests issued from the web browser to web server application 910 hosted by camera 100. By implementing remote accessibility via the internet 750, access to camera 100 can be obtained from virtually an unlimited number of locations. For example, camera 100 can be set to continuously take pictures of scenes/items of interest and allow user 720 to access those pictures at any time. Camera 100 and web server application 910 hosted thereon are further described in the discussion of FIG. 9 below.

Referring still to FIG. 7, it should be appreciated that camera 100, in accordance with the present invention, does not require a separate, external computer system (e.g., a personal computer) for connecting to ISP 710, thus providing an inexpensive method for making remotely accessible cameras widely available. It should be further appreciated that while process 700 shows camera 100 coupling to internet 750 via one ISP (e.g., ISP 710) and user 720 coupling to internet 750 via a separate ISP (e.g., ISP 715), user 720 and camera 100 could be coupled to internet 750 through a single ISP. In such a case, user 720 and camera 100 would be coupled to two separate access ports (e.g., two separate modems out of a bank of modems) of the same ISP.

In addition, camera 100 is accessed via the widely used, very familiar web browser. By functioning with a web page based interface and widely used web browsers, the present invention provides a simple, intuitive, and familiar interface for accessing camera 100's functionality. Accordingly, camera 100's controls and functions are intuitively easy to utilize. Since web pages and their associated controls (e.g., push buttons, data entry fields, etc.) are very familiar to most users, the remote access functionality of camera 100 can be utilized without requiring a extensive learning period for new users. For example, a consumer purchasing a remotely accessible camera is typically able to easily and immediately use the remote accessibility functions with minimal set-up.

As described above, the remote accessibility of camera 100 provides for many new applications of digital imagery. One such application involves setting up camera 100 at some remote location and using it to take pictures at successive intervals. These pictures would be accessed via the internet 750 as they are taken. The interval can be adjusted (e.g., more or less pictures per minute) in response to user 720 entered commands via a Web browser. In such an application the limited memory (e.g., DRAM 346 and removable memory 354 of FIG. 3) of camera 100 would be used to hold a desired number of pictures at a specified

resolution. The memory would function as a sort of FIFO, wherein a fixed number of pictures are stored, the latest picture replacing the earliest picture.

Another application involves using camera 100 in conjunction with a motion detector. When used in conjunction with a motion detector, camera 100 can be configured to capture an image in response to receiving a signal from the motion detector (e.g., detecting the motion of an intruder), thereby taking a picture of whatever triggered the detector's signal output. Alternatively, camera 100 can detect motion by simply comparing successive images to detect changes between them, thereby dispensing with the need for a separate motion detector. The camera can additionally be configured to notify user 720 (e.g., via an email) to access and view the image of the potential intruder.

Yet another application involves using camera 100 in conjunction with a remote aiming device. Camera 100 can be mounted on a remotely operated aiming device (e.g., a motorized tripod). The aiming device is controlled via the internet 750 in the same manner the camera is controlled via the internet 750. Alternatively, camera 100 could be coupled to control the remote aiming device directly, via a software routine executing on computer 118 (shown in FIG. 1). The remote aiming device allows user 720 to control the field of view of the camera 100 in the same manner user 720 controls other functionality (e.g., picture resolution, picture interval, etc.). User 720 can position the camera to take pictures of objects in the camera's vicinity.

In this manner, system 700 of the present invention is able to implement sophisticated remote surveillance of the type previously performed by expensive, prior art closed circuit television devices. Unlike the prior art, however, system 700 is inexpensive and relatively simple to implement.

For example, to achieve the same functionality as system 700 with a prior art personal computer in place of camera 100, custom-designed software would have to be written to host a web server on the personal computer. This software would have to function in conjunction with the operating system software of the personal computer. This, in turn, leads to a large amount of complexity and difficulty configuring and maintaining the personal computer/software. In addition to the personal computer, an external imaging device would also be required. Hence, the personal computer becomes a very expensive, dedicated platform for hosting the web server.

In contrast, camera 100 of system 700 includes the necessary software and the necessary computational resources (e.g., computer 118) to host the web page itself, eliminating the requirement for the expensive personal computer. In so doing, remote viewing, remote surveillance, a remote picture taking operations are made much more usable and much more obtainable to the average user. This "web site enabled" camera greatly reduces the cost of achieving the above functionality. The reduced cost will lead to wide adoption and deployment of the present invention, which will in turn, lead to a large number of new applications and new software written to take advantage of the resulting installed based of low-cost internet enabled, remotely accessible cameras of the present invention.

Referring now to FIG. 8, a more detailed diagram 800 of camera 100 coupled to internet 750 is shown. Diagram 800 shows camera 100 coupled to an external modem 801. Camera 100 is coupled to modem 801 via any of several communications means (e.g., USB, IEEE1394, infrared link, etc.). Modem 801 is in turn coupled to a POTS telephone jack 802 at the camera's location. The telephone jack 802

couples modem 801 to one of the modems 803 of ISP 710 via the telephone companies local loop, ISP 760, as described above, is directly coupled to the internet 750 via a T1 line.

Modem 801 is shown as an external modem. However, the functionality of modem 801 can be implemented directly within the electronics of camera 100 (e.g., via a modem ASIC), or alternatively, can be implemented as a software only modem executing on computer 118 within camera 100. As such, it should be appreciated that, at the hardware connectivity level, modem 801 can take several forms. For example, a wireless modem can be used in which case the camera is not connected via an external wire to any land line. Alternatively, there may even be applications in which camera 100 includes suitable electronic components enabling a connection to a conventional computer system network (e.g., ethernet, Apple talk, etc.), which is in turn, directly connected to the internet (e.g., via a gateway, a firewall, etc.), thereby doing away with the requirement for an ISP. Hence, it should be appreciated that the present invention is not limited to any particular method of accessing the internet 750.

Referring now to FIG. 9, a diagram 900 of the connectivity and application software of camera 100 is shown. At the software level, computer 118 of camera 100 hosts a TCP/IP protocol stack 901 (including PPP (Point to Point Protocol)), which, as is well-known in the art, enables communication via the internet. Protocol stack 901 interfaces with the physical connection hardware 902 of camera 100 and the application layer 903. The bottom of protocol stack 901 includes communication hardware interface drivers which interfaces directly with the various communications hardware camera 100 must function with (e.g., USB, IEEE1394, etc.). The top of protocol stack 901 includes software APIs and protocol libraries which interface with web server application 910 running in an applications layer 903. Applications layer 903 interfaces with an operating system 904. Applications layer 903, protocol stack 901, and operating system 904 are instantiated as software modules in DRAM 346 of camera 100.

The web server application 910 runs within applications layer 903, along with other software applications which provide camera 100's functionality (e.g., still image downloading, motion detection, aim control for a remote aiming device, and the like). The web server application 910 responds to queries from the user's internet web browser and other web browsers, which include user requests and user commands directed to the camera (e.g., taking the picture, changing the picture taking interval, etc.) and communicates with other software applications within applications layer 903. These applications each communicate with operating system 904 of the camera 100, which controls the functionality of camera 100 (e.g., taking pictures, storing pictures, and the like). HTTP requests are received and HTML files are transferred to and from the web server application 910 via protocol stack 901, and communications hardware 902.

With reference now to FIG. 10, a more detailed diagram of ID server 760 is shown. As described above, ID server 760, in accordance with the present invention, overcomes the unknown address problem, wherein the internet address of camera 100 changes each time it establishes a connection. ID server 760 solves this problem by maintaining a registry 1001 of relevant internet addresses. Registry 1001 solves the unknown address problem by cataloging a device's unique identifier (e.g., camera 100), the address of the device (e.g., address 1002), and any relevant user information (e.g., user info 1003).

13

In the present embodiment, the registry is a software data structure residing in ID server 760. ID server 760 has an internet address which is known by both the device and the user. When camera 100 is connected to the internet 750, camera 100 accesses ID server 760, notifying ID server 760 that it is now "on-line". Camera 100 informs ID server 760 of its current internet address. ID server 760 updates registry 1001 accordingly. As described above, this internet address is different for each time camera 100 is connected to the internet 750. Each time ID server 760 is notified that the digital camera is online, the registry 1001 is updated with camera 100's current internet address.

Subsequently, when user 720 attempts access to camera 100, user 720's web browser first accesses ID server 760. If camera 100 is on-line, ID server 760 finds a corresponding entry in registry 1001, and returns the correct address (e.g., address 1002) to the web browser. Using the address, the web browser subsequently accesses camera 100. If camera 100 is not on-line, the web browser notifies user 720, by, for example, displaying an appropriate message (e.g., DNS entry not found). This process can be implemented in a manner which is transparent to user 720. For example, once camera 100 is registered within registry 1001, user 720's web browser can access ID server 760 automatically to obtain the "current" internet address camera 100 and subsequently access camera 100 in a single user step (e.g., by clicking on a "bookmark" associated with the camera or by "typing in" the camera's identifier).

Similarly, ID server 760 can also maintain user 720's email address within registry 1001 (e.g., within user info 1003). In so doing, camera 100 will be able to notify user 720 of preprogrammed events (e.g., a camera malfunction, such as low battery power) or other such information by sending an appropriate email message.

It should be appreciated that there are many variations of service possible using this scheme. For example, camera 100 can be configured to automatically notify user 720 via ID server 760 when camera 100 is on-line. Alternatively, ID server 760 may itself notify user 720 when it is on-line with new pictures for viewing (e.g., via a page or telephone call). Any particular variation can be implemented depending upon the particular functionality desired by a user.

For example, security can be enhanced by maintaining a system of passwords between camera 100 and user 720. When user 720 accesses ID server 760, user 720 might be required, for example, to enter appropriate user authorization information, such as, for example, a user ID and password. Similarly, when camera 100 accesses ID server 760, it might also be required to provide a "device ID" and a password. Using this information, ID server 760 can ensure only authorized users are coupled to authorized devices.

Alternatively, security can be maintained by camera 100 in addition to, or instead of, ID server 760. Once camera 100 has notified the ID server 760 that it is on-line, it services all of requests for access. However, full access (e.g., access to the functionality of the camera) is accorded only to those users having appropriate authorization information (e.g., user ID and password). Whereas unauthorized user attempting access might receive an appropriate message (e.g., an "access denied" web page), an authorized user would see a web page representative of the functionality of the camera. The web page could include, for example, control buttons for camera control, images, or the like.

It should be noted that the first time ID server 760 is accessed, user 720 may be prompted to enter appropriate

14

information (e.g. device ID, password information, etc.) to initialize and set up the service. This information uniquely identifies both user 720 and camera 100. The initialization can be made completely automatic beyond user 720 entering the appropriate information. Once the initialization process is completed, the operation of user 720's web browser with ID server 760 would proceed transparently with respect to user 720.

With reference now to FIG. 11, a flow chart of a process 1100 in accordance with one embodiment of the present invention is shown. Process 1100 shows the steps of an operating process of remotely accessing images taken by and stored within a digital camera of the present invention (e.g., camera 100) by accessing a web page hosted by the digital camera.

Process 1100 begins in step 1101, where a digital camera in accordance with one embodiment of the present invention is coupled to the telephone system. As described above, the digital camera couples to the telephone system via either an internal hardware modem, an internal software based modem, or an external modem. In this manner, the digital camera connects to the telephone system directly, such that a separate, dedicated computer system (e.g., a personal computer) is unnecessary.

In step 1102, the digital camera determines whether it has been previously initialized. As described above, if the digital camera has been previously initialized, the appropriate identification and password information has been previously entered by the user such that, in the present embodiment, the connection to the internet is entirely automatic. The camera uses a dial up connection from the user's ISP to access the internet and process 1100 proceeds to step 1104. If the digital camera has not been previously initialized (e.g., as is the case when the digital camera is newly purchased by the user), the communications routines used by digital camera need to be specified by the user and process 1100 proceeds to step 1103. These routines includes specific information the camera needs to connect to the internet (e.g., dialing prefixes, ISP phone number, passwords, and the like).

In step 1103, the digital camera is initialized with appropriate connectivity information. This information includes for example, dialing prefixes, ISP connection information, passwords, user ID information, device ID information, and the like. This information enables the digital camera to automatically connect to the internet, as needed. As described above, in the case where the digital camera is coupled to the internet via some other means, (e.g., ethernet connection via a firewall) the connectivity information changes accordingly.

In step 1104, once connected to the internet, the digital camera notifies the ID server 760 that is currently on-line. As described above, this involves registering its current internet address with the ID server 760.

In step 1105, the application programming within the digital camera implements the user's application. As described above, the system of the present invention is capable of implementing a wide variety of remote access, remote imaging/surveillance applications. In the present embodiment, the digital camera merely records successive images for remote access by the user. The images are loaded into the camera's memory on a FIFO basis, with the earliest recorded image being replaced by the latest recorded image. The number of images available to the user depends upon the amount of installed memory in the camera. The digital camera periodically accesses the internet via the ISP (e.g., at the top of every hour) to allow the user to access and retrieve the stored images.

15

In step 1106, the user, or another "web surfer", accesses the ID server 760 with a web browser to retrieve the internet address of the digital camera. As described above, the ID server 760 solves the unknown address problem, wherein the digital camera receives a different internet address from the ISP each time it connects to the internet. As described above, the user enters the identity of the digital camera into his web browser (e.g., the camera's URL). Using standard internet protocols, ID server 760 is queried with the URL, and returns the digital camera's current internet address.

In step 1107, the user's web browser then accesses the digital camera using the camera's current internet address returned from ID server 760. As described above, the digital camera includes the necessary computer resources to function as a web site and host its own internal web server application 910. When accessed by the user's browser, the digital camera transmits HTML (hyper text mark-up language) document files for its web page. As described above, different web pages can be shown to different accessing web browsers depending upon their authorization (e.g., via passwords). For example, as described above, an unauthorized user might receive a web page displaying an "access denied" sign.

In step 1108, the user accesses the images stored within the digital camera via the web pages received from the web server application 910 hosted within the digital camera. As described above, the web page interface of the digital camera provides a readily familiar and intuitive interface for interaction and control of the camera by the user. Depending upon the particular application, the camera's web pages include control buttons, data entry fields, drop down menus, or even more sophisticated objects (e.g., java applets) for interaction with the user. Using these web pages, the user is able to access the functional controls of the camera in addition to the stored images.

In step 1109, process 1100 continues depending upon the particular requirements of the user. For example, as described above, the user can modify the parameters of the application program executing within the camera (e.g., increase or decrease the frequency of image recording). The user can let the application continue running as is. The virtually zero incremental cost of the images allows for many variations.

Thus, the present invention provides a method for making a digital camera and its internally stored images remotely accessible. The present invention enables the digital camera to be set to continuously take pictures of scenes/items of interest and allow a user to access those pictures at any time. The present invention implements remote accessibility via the internet, thus allowing the user to access the digital camera from virtually an unlimited number of locations.

A digital camera in accordance with the present invention does not require an separate, external computer system (e.g., a personal computer) for internet connectivity, thus providing an inexpensive method for making remotely accessible digital cameras widely available. In addition, a digital camera in accordance with the present invention is accessed via the widely used, very familiar web browser. By functioning with typical, widely used web browsers, the present invention provides a simple, intuitive, and familiar interface for accessing the digital camera's functionality. In so doing, the controls and functions of the digital camera are intuitively easy to utilize, and do not require an extensive learning period for new users. The present invention also provides an efficient, user transparent, process of obtaining the internet address of a digital camera, where the camera accesses the

16

internet via a dial-up connection, and thus, has a changing internet address.

The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.

What is claimed is:

1. A method for implementing internet access to images stored in a portable digital image capture unit, wherein the portable image capture unit includes a processor coupled to a memory, the method comprising the steps of:

- a) directly capturing an image using the portable image capture unit;
- b) establishing a direct internet connection by the portable image capture unit without requiring a computer system for internet connectivity, wherein a current internet address of the portable image capture unit changes each time the portable image capture unit connects to the internet, wherein the portable image capture unit hosts a web page;
- c) accessing an ID server and registering the current internet address of the portable image capture unit;
- d) accessing the ID server via the internet with a web browser of a user;
- e) retrieving the current internet address of the portable image capture unit from the ID server transparently to the user; and
- f) using the current internet address and the web browser to access the web page hosted by the portable image capture unit in order to access the image from the portable image capture unit.

2. The method of claim 1 wherein step b) further includes the step of coupling the portable image capture unit to the internet via a direct telephone dial-up internet connection.

3. The method of claim 1 further including the step of initializing the portable image capture unit with connectivity information which enables the establishment of a direct dial-up internet connection.

4. The method of claim 1 further including the step of communication authorization information between the portable image capture unit and the web server.

5. The method of claim 1 further including the step of communicating authorization information between the portable image capture unit and the web browser.

6. The method of claim 1 further including the portable image capture unit performing the step of sending an electronic mail message to the user when the portable image capture unit contains the image.

7. The method of claim 1 further including the step of manipulating an object included in the web page to access the image.

8. The method of claim 1 further including the step of manipulating an object included in the web page to access control functions of the portable image capture unit.

9. The method of claim 1 wherein the portable image capture unit is a hand held digital camera.

10. The method of claim 1 wherein the ID server includes the functionality of a domain name server.

17

11. A system for remotely accessing images stored in a digital portable image capture unit, wherein the portable image capture unit includes a processor coupled to a memory, the memory containing computer readable code, which when executed by the processor cause the portable image capture unit to implement a method, the method comprising the steps of:

- a) directly capturing an image;
- b) accessing a web server via a direct connection with the internet without requiring a computer system for internet connectivity, wherein a current internet address of the portable image capture unit changes each time the image capture unit accesses the web server via the internet;
- c) registering the current internet address of the portable image capture unit with the web server;
- d) hosting a web page such that the image can be accessed by a web browser of a user wherein the web browser obtains the current internet address from the web server transparently to the user.

12. The method of claim 11 wherein step b) further includes the step of coupling to the internet via a direct telephone dial-up internet connection.

13. The method of claim 11 further including the step of storing initialization information enabling the establishment of a direct telephone dial-up internet connection.

14. The method of claim 11 further including the step of communicating authorization information to the web server.

15. The method of claim 11 further including the step of authenticating the user prior to providing the web page to the user.

16. The method of claim 11 further including the portable image capture unit performing the step of sending an electronic mail message to the user when the portable image capture unit contains the image.

17. The method of claim 11 wherein the image capture unit is a digital camera.

18

18. The method of claim 11 wherein the web server is a domain name server.

19. In a portable image capture unit including a processor coupled to a memory, the memory containing computer readable code, which when executed by the processor cause the portable image capture unit to implement a method for providing remote access to an image stored in the portable image capture unit, the method comprising the portable image capture unit performing the steps of:

- a) directly capturing an image;
- b) coupling directly to the internet via a telephone dial-up internet connection without requiring a computer system for internet connectivity;
- c) storing initialization information enabling the establishment of the direct internet connection;
- d) accessing a web server via the direct connection to the internet, wherein a current internet address of the portable image capture unit changes each time the portable image capture unit accesses the web server via the direct connection to the internet;
- e) registering the current internet address of the portable image capture unit with the web server; and
- f) hosting a web page such that the image can be accessed by a web browser of a user, wherein the web browser obtains the current internet address from the web server transparently to the user.

20. The method of claim 19 further including the step of communicating authorization information to the web server.

21. The method of claim 19 further including the step of authenticating the user prior to providing the web page to the user.

22. The method of claim 19 further including the steps of: retrieving the internet address of the web browser of the user from the web server; and contacting the user when the portable image capture unit contains images.

* * * * *



US006784924B2

(12) United States Patent
Ward et al.**(10) Patent No.: US 6,784,924 B2****(45) Date of Patent: Aug. 31, 2004****(54) NETWORK CONFIGURATION FILE FOR
AUTOMATICALLY TRANSMITTING
IMAGES FROM AN ELECTRONIC STILL
CAMERA****(75) Inventors:** Joseph Ward, Hilton, NY (US);
Kenneth A. Parulski, Rochester, NY
(US); James D. Allen, Rochester, NY
(US)**(73) Assignee:** Eastman Kodak Company, Rochester,
NY (US)**(*) Notice:** Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 0 days.**(21) Appl. No.: 09/004,046****(22) Filed: Jan. 7, 1998****(65) Prior Publication Data**

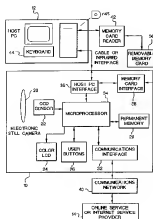
US 2003/0142215 A1 Jul. 31, 2003

Related U.S. Application Data**(60)** Provisional application No. 60/037,963, filed on Feb. 20,
1997, and provisional application No. 60/037,962, filed on
Feb. 20, 1997.**(51) Int. Cl.** **H04N 5/225****(52) U.S. CL.** **348/207.1; 348/211.3****(58) Field of Search** **348/207, 211,**
348/212, 213, 231, 232, 233, 552, 239,
211 99-211.13, 211.3, 17; 710/62, 3, 8;
345/327; 386/48, 117; 725/131-133, 139-141,
151-153, 709/217-219, 220, 705/26, 27,
455/566**(56) References Cited****U.S. PATENT DOCUMENTS**4,524,381 A 6/1985 Konishi
4,574,319 A 3/1986 Konishi
4,825,457 A 4/1989 Lebowitz
5,241,659 A 8/1993 Parulski et al.
5,477,264 A 12/1995 Sarbadhikari et al.5,606,365 A 2/1997 Maurinus et al.
5,663,678 A 9/1997 Chang
5,666,159 A 9/1997 Parulski et al.
5,737,491 A * 4/1998 Allen et al. 704/270
5,800,005 A * 9/1998 Hull et al. 455/566
5,825,432 A * 10/1998 Yonezawa 348/563
6,111,604 A * 8/2000 Hashimoto et al. 348/220
6,122,005 A * 9/2000 Sasaki et al. 348/211.3
6,166,729 A * 12/2000 Acosta et al. 345/719
6,449,001 B1 * 9/2002 Levy et al. 348/1404**OTHER PUBLICATIONS**User's Manual—Axis 2420 Network Camera, 1996 (Axis
Communications).*U.S. patent application Ser. No. 60/037,963, Parulski et al.
“The Universe of Smart Cards: EasySIM software”,
Schlumberger Limited.

“Claris Emailer” from Claris’ home page on internet.

“Kodak Professional Digital Camera System—User’s
Manual,” Eastman Kodak Company, 1991–1992, pp. x to xv,
2–1 to 2–3, 3–29 to 3–30, and 4–1 to 4–49.

* cited by examiner

Primary Examiner—Tuan Ho**(74) Attorney, Agent, or Firm—Pamela R. Crocker****(57) ABSTRACT**A network configuration file is generated at a host computer
and downloaded to a digital camera. This file contains
instruction information for communicating with a selected
destination via a communications interface. The digital
camera includes a “send” button or LCD icon which allows
the user to easily transmit one or more images via a wired
or wireless communications interface to a desired
destination, which among other possibilities may be an
Internet Service Provider or a digital photofinishing center.
When the user selects this option, the communications port
settings, user account specifics, and destination connection
commands are read from the network configuration file on
the removable memory card. Examples of these settings
include serial port baud rate, parity, and stop bits, as well as
account name and password.**9 Claims, 4 Drawing Sheets**

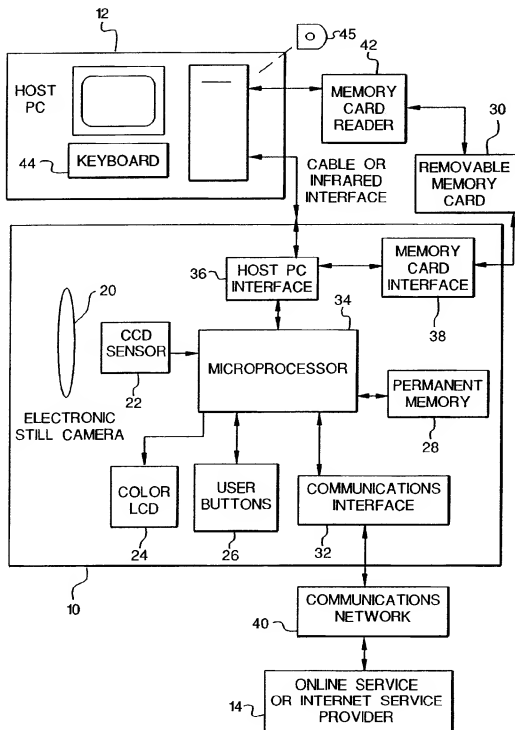
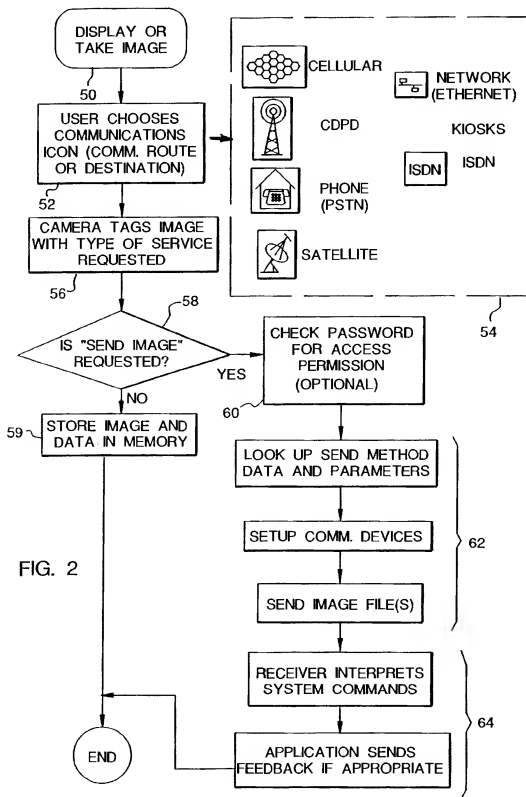


FIG. 1



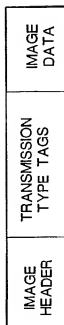


FIG. 3



FIG. 4

FIG. 4A

PHONE (PUBLIC SWITCHED TELEPHONE NETWORK)



PROTOCOL TYPE	PHONE NUMBER	DEFAULT SETTINGS	MODERN CONTROL STRING	ACCOUNT DATA	PASSWORD	SYSTEM COMMANDS
------------------	-----------------	---------------------	-----------------------------	-----------------	----------	--------------------

CELLULAR OR PCS

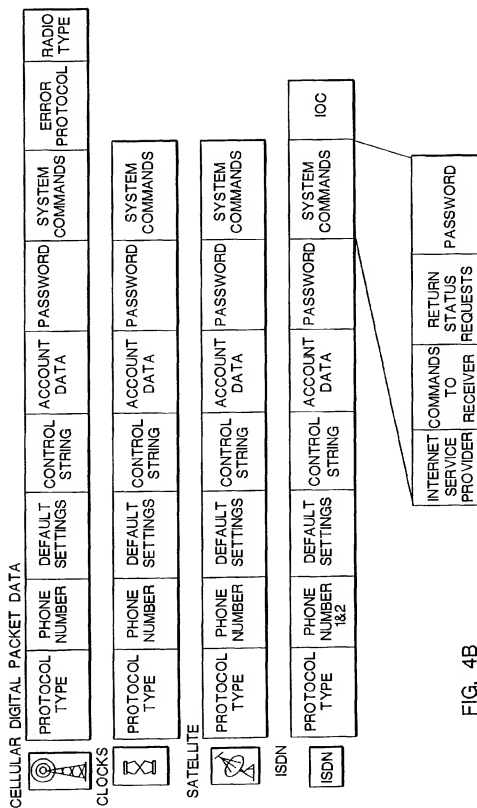


PROTOCOL TYPE	PHONE NUMBER	DEFAULT SETTINGS	MODERN CONTROL STRING	ACCOUNT DATA	PASSWORD	SYSTEM COMMANDS	ERROR PROTOCOL	RADIO TYPE
------------------	-----------------	---------------------	-----------------------------	-----------------	----------	--------------------	-------------------	---------------

WIRELESS LAN



PROTOCOL TYPE	PHONE NUMBER	DEFAULT SETTINGS	PARA- METER FILE	ACCOUNT DATA	PASSWORD	SYSTEM COMMANDS
------------------	-----------------	---------------------	------------------------	-----------------	----------	--------------------



1

NETWORK CONFIGURATION FILE FOR AUTOMATICALLY TRANSMITTING IMAGES FROM AN ELECTRONIC STILL CAMERA

CROSS-REFERENCE TO RELATED APPLICATION(S)

THIS APPLICATION CLAIMS BENEFIT OF PROVI-
SIONAL APPLICATION SERIAL NO. 60/037,962 FILED
Feb. 20, 1997.

Reference is made to commonly assigned copending
applications Serial No. 60/037,963, filed Feb. 20, 1997
entitled "Electronic Camera with 'Utilization' Selection
Capability" and filed in the names of Kenneth A. Parulski,
Joseph Ward, and Michael C. Hopwood, which is assigned
to the assignee of this application.

FIELD OF THE INVENTION

The invention relates generally to the field of
photography, and in particular to electronic photography.
More specifically, the invention relates to a digital camera
that interfaces with a host computer.

BACKGROUND OF THE INVENTION

Digital cameras, such as the Kodak Digital Science
DC25™ camera, allow images to be utilized on a home
computer (PC) and to be incorporated into e-mail documents
and personal home pages on the World Wide Web. Presently,
images must be copied to the PC and transmitted as e-mail,
for example using an online service or an Internet Service
Provider (ISP), via a modem from the user's PC. It would be
desirable to be able to transmit pictures directly from the
digital camera instead of first transferring the pictures to a
PC. For instance, on a vacation trip, it is desirable to
immediately share pictures with friends or relatives via
e-mail or Internet access. It is also desirable to transmit
pictures from a location without PC access in order to free
up camera storage to take additional pictures. There are a
wide variety of connection means to online services such as
America On Line, ISPs, and bulletin board services. Each of
these services typically requires an account name and
password, as well as local telephone access numbers, and
specific communications settings. It would be difficult to
provide an easy-to-use means with buttons or menus on a
small digital camera to input and/or modify all of these
required settings.

SUMMARY OF THE INVENTION

The present invention is directed to overcoming one or
more of the problems set forth above. Briefly summarized,
according to one aspect of the present invention, a network
configuration file is generated at a host computer and
downloaded to a digital camera. This file contains instruction
information for communicating with a selected destina-
tion via a communications interface. The digital camera
includes a "send" button or LCD icon which allows the user
to easily transmit one or more images via a wired or wireless
communications interface to a desired destination, which
among other possibilities may be an Internet Service Pro-
vider or a digital photofinishing center. When the user
selects this option, the communications port settings, user
account specifics, and destination connection commands are
read from the network configuration file. Examples of these
settings include serial port baud rate, parity, and stop bits, as
well as account name and password.

2

In addition, information about which image or images to
transmit is entered using the user buttons on the digital
camera. This information is used to automatically establish
a connection, log-in to the desired destination, and to
transmit the image. The transmission may occur immedi-
ately after the pictures are taken, for example if the camera
has a built-in cellular phone modem, or at a later time, when
the camera is connected to a separate unit (such as a dock,
kiosk, PC, etc.) equipped with a modem. In the latter case,
a "utilization file" is created to provide information on
which images should be transmitted to which account.

These and other aspects, objects, features and advantages
of the present invention will be more clearly understood and
appreciated from a review of the following detailed descrip-
tion of the preferred embodiments and appended claims, and
by reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a system block diagram of the invention.
FIG. 2 is a diagram showing the steps used to automati-
cally transmit images using the network configuration file.
FIG. 3 is a diagram of an image file.
FIG. 4 is a diagram showing several versions of the
network configuration file.

DETAILED DESCRIPTION OF THE INVENTION

Because imaging systems and devices are well known, the
present description will be directed in particular to elements
forming part of, or cooperating more directly with, apparatus
in accordance with the present invention. Elements not
specifically shown or described herein may be selected from
those known in the art. Some aspects of the present descrip-
tion may be implemented in software. Unless otherwise
specified, all software implementation is conventional and
within the ordinary skill in the programming arts.

A system block diagram of the invention is shown in FIG.
1 including an electronic still camera 10, a host computer
(PC) 12 and a service provider 14. The camera includes an
optical section 20 for imaging a scene upon a CCD sensor
22 and generating an image signal, a liquid crystal display
(LCD) 24 for displaying images and other information, a
number of user input buttons 26, both permanent memory 28
and removable memory 30, and an internal communications
interface 32 (e.g., modem). This interface may connect to a
variety of known networks, such as a public switched
telephone network (PSTN), ISDN, an RF cellular phone
network, or Ethernet. The camera 10 also includes a micro-
processor 34 for generally controlling the camera functions,
as well as the interchange of data with the host PC 12 and
the memory card 30 through a host PC interface 36 and a
memory card interface 38, respectively. Besides the host PC
12, the system includes a network connection 40 to the
online service or ISP (Internet Service Provider) 14.
Alternately, the network 40 can connect to the user's home
PC 12.

When the camera 10 is first purchased (or at any time
thereafter), it is connected to the PC 12 via the host PC 36
interface and a software application (stored on a disc 45)
running on the host PC 12 will enable the user to specify the
name of a destination ISP or online service and to input from
the host PC keyboard 44 the appropriate communication
settings and account information. This information gener-
ates a network configuration file, which then can be then be
downloaded to the camera 10 through the host PC interface

3

4

36, which may be a wired or infrared (e.g., IrDA) interface, and written to the camera's internal memory 28 and/or the removable memory card 30. Alternatively, a host PC equipped with a memory card reader/writer 42 can write the information directly to the card 30 without connecting the camera through its host PC interface 36. Also, this information could be predetermined by the user and stored in a "preferences" file on the host PC 12 and then transferred to the camera 10 from this file without further intervention by the user. Multiple sets of destination services can be stored on the memory card 30. Typically, keyword or graphic descriptors (e.g., icons) accompany the information in the network configuration file about destination services to enable easy access by the camera user.

The steps used to automatically transmit images using the network configuration file are shown in FIG. 2. After disconnecting the camera from the host PC, the user operates the camera to take pictures (step 50). This is typically done at a remote location, for example while traveling to another city. As the user takes or reviews images on the image LCD display, the decision can be made to transmit one or more images (step 52). This is done by choosing one of the keywords or icons in a menu 54 shown in FIG. 2, which are displayed on the LCD 24 and selected, e.g., through the user buttons 26. (Note that a camera will typically only include a subset (only those desired by the user) of all the different services shown.) The selected image files may be tagged with a code (step 56) indicating which service is requested, as shown in FIG. 3. (Alternately, an "image utilization" file can be created in the camera storing a list of images to be transmitted by a particular method, as described in the cross-referenced depending patent application (U.S. Serial No. 60/037,963). As described in that patent application, the details of an order, e.g., number of print copies to be made from an image and the size of the prints and/or a list of images to be e-mailed to various recipients, is written into the "utilization" file, which identifies the order and includes pointers to the image files that store the images required to "fulfill" the order. The "utilization" file is stored in the internal memory 28 or the memory card 30.)

Next, the system determines whether a request exists to send an image (step 58). If no request is present, the image and associated data is stored in either permanent memory 28 or the memory card 30 (step 59). (Typically, all images are initially saved in memory whether eventually sent or not.) Otherwise, if there is a request to send an image, the user ensures that the camera is connected to the appropriate service (wired telephone line, cellular phone, kiosk, etc.) and pushes a "send" button in the user button section 26, or selects a "send" menu option on the LCD 24. The camera then utilizes the appropriate network configuration file, shown in FIG. 4. Each network configuration file contains items such as the protocol type, phone number, etc., as described in Appendix I. The user password may be checked against the password in the network configuration file to ensure that the user is authorized to connect the camera to the desired service (step 60). Alternately, the stored password in the appropriate configuration file can be used. Next, the camera uses the parameters in the configuration file to establish communications with the service and send one or more image files as selected by the user (steps 62). The service receiver interprets the system commands issued by the camera from the network configuration file list and sends appropriate feedback (such as "transfer in progress" and "transfer complete") which are interpreted by the camera and displayed on the LCD 24 (steps 64).

For example, when the camera uses a normal wired telephone (Public Switched Telephone Network) connection

(i.e., network 40) to the camera's internal modem 32, after the user selects the images to be sent and presses the "send" button, the camera performs the following steps without user intervention:

- 1) Read the appropriate connection parameters from the network configuration file (on the memory card 30 or internal camera memory 28), dial the phone and establish the connection to the destination service 14.
- 2) Read the user's account name and password and transmit these to "log-on" to the service 14.
- 3) Using the appropriate communications protocol (FTP, mailto, etc.), transmit the selected image or images to the destination service 14.

The invention has been described with reference to a preferred embodiment. However, it will be appreciated that variations and modifications can be effected by a person of ordinary skill in the art without departing from the scope of the invention.

Appendix I

These are descriptions of the tags listed in the previous drawing:

Protocol Type

Each communication method has its own protocol, or rules to communicate. This tag identifies that protocol and where to find it. For example, the Network may use TCP/IP and a modem may use XModem.

Phone Number

This is the number of the receiving service. If internet access is requested, this could be the number of the Internet Service Provider. For ISDN, some systems require two phone numbers, dialed and connected to in sequence.

Default Settings

Standard settings that make the communications device compatible with the imaging device.

Modem Control String

Modem and communications devices have a command language that can set them up before they are used. For example, modems have many options controlled by command strings including volume level, the amount of time the carrier is allowed to fail before the system hangs up, and so on.

Account Data

This can be internet account data, charge number data, phone card data, billing address, and data related to the commerce part of the transmission.

Password

Any password needed to get into the communications system. Other passwords to get into the remote application or destination are located in the System Commands section.

System Commands

These are commands that control the end destination.

Error Protocol

In cellular and some other wireless communications, error protocols are used to increase the robustness of the link. For example, MNP10 or ETC may be used for cellular links.

Radio Type

The type of radio used for this communications feature may be identified here. Some cell phones have modems built in, others will have protocols for many communications functions built in. The radio type will make the imaging device adapt to the correct interface.

IOC

ISDN Ordering Code identifies what features are available on the ISDN line provided by the telco. It is used to establish the feature set for that communications link.

5

Internet Service Provider

This identifies the actual service provider and any specific information or sequence of information that the service wants to see during connection and logoff. It also tells the device how to handle the return messages, like "time used" 5 that are returned by the server.

Commands to Receiver

This may be a list of commands to control the receiving application. For example, a command to print one of the images and save the data to a particular file on a PC may be embedded here. 10

Return Status Requests

This tag can set up the ability of the application to tell if an error has occurred, or what the status of the application might be. The data here will help the device decide if it should continue communicating and a set user interface response can be developed around this feedback. 15

What is claimed is:

1. A method for directing the automatic transmission of images from an electronic camera, said method including the steps of: 20

storing network configuration information in a memory of the electronic camera;

using the network configuration information to connect the camera to an online service or internet service provider (ISP); 25

transferring pictures and data from the camera to the online service or ISP;

receiving information from the online service or ISP indicating that the transfer of the pictures and data is complete; and 30

displaying information on a display of the electronic camera indicating that the transfer of the pictures and data is complete; 35

wherein the network configuration information is generated at least in part in a host device adaptable for communication with the electronic camera, based on information previously stored in the host device and utilizable to connect the host device to the online service or ISP. 40

2. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes a PSTN (Public Switched Telephone Network).

3. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes an ISDN network. 45

4. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes an RF cellular phone network.

6

5. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes an Ethernet connection.

6. The method of claim 1 wherein the information previously stored in the host device comprises information stored in a network configuration preference file of the host device.

7. A method for directing the automatic transmission of images from an electronic camera, said method including the steps of:

storing network configuration information in a memory of the electronic camera, the network configuration information comprising information for using a communications network to establish a connection between the electronic camera and an online service or internet service provider (ISP); and

in response to user activation in the electronic camera of a transmission command specifying one or more pictures for transmission, automatically performing without further user intervention the steps of:

retrieving the stored network configuration information;

utilizing the retrieved information to establish a connection between the electronic camera and the online service or ISP; 25

transferring the one or more pictures from the electronic camera to the online service or ISP;

receiving information from the online service or ISP indicating that the transfer of the one or more pictures is complete; and

displaying the information on a display of the electronic camera indicating that the transfer of the one or more pictures is complete; 30

wherein the network configuration information is generated at least in part in a host device based on information previously stored in the host device, and the generated network configuration information is subsequently downloaded from the host device into the electronic camera in conjunction with the storing step. 35

8. The method of claim 7 wherein the host device comprises a personal computer.

9. The method of claim 7 wherein the information previously stored in the host device comprises information stored in a network configuration preference file of the host device. 40

* * * * *



US006636259B1

(12) **United States Patent**
Anderson et al.

(10) **Patent No.:** **US 6,636,259 B1**
(45) Date of Patent: **Oct. 21, 2003**

(54) **AUTOMATICALLY CONFIGURING A WEB-ENABLED DIGITAL CAMERA TO ACCESS THE INTERNET**

6,167,469 A * 12/2000 Safai et al. 710/62
 6,226,752 B1 * 5/2001 Gupta et al. 713/201
 6,269,481 B1 * 7/2001 Perlman et al. 717/11
 6,502,195 B1 * 12/2002 Colvin 713/202

(75) Inventors: **Eric C. Anderson**, San Jose, CA (US);
Robert Paul Morris, Raleigh, NC (US)

* cited by examiner

(73) Assignee: **IPAC Acquisition Subsidiary I, LLC**,
 Peterborough, NH (US)

Primary Examiner—Wendy R. Garber
Assistant Examiner—Jacqueline Wilson

(74) *Attorney, Agent, or Firm*—Sawyer Law Group LLP

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 32 days.

(57) **ABSTRACT**

(21) Appl. No.: **09/625,824**

(22) Filed: **Jul. 26, 2000**

(51) **Int. Cl.**⁷ **H04N 5/232**

(52) **U.S. Cl.** **348/211.3; 348/211.12; 348/14.04**

(58) **Field of Search** **348/14.01, 14.02, 348/14.04, 14.08, 14.09, 552, 143, 156, 157, 211.3, 211.12; 709/322**

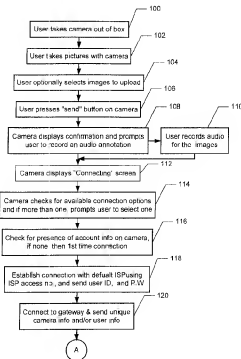
(56) **References Cited**

U.S. PATENT DOCUMENTS

5,430,827 A * 7/1995 Rissanen 704/272
 5,737,491 A * 4/1998 Allen et al. 704/270
 5,905,736 A * 5/1999 Ronen et al. 370/546
 6,064,671 A * 5/2000 Killian 370/389
 6,067,571 A * 5/2000 Igarashi et al. 709/232

The present invention provides a method and system for automatically establishing a user account and for configuring a hand-held electronic device for accessing a site on a public network using the user account. The method and system include establishing a connection to a website server, which is responsible for establishing and maintaining website accounts. The presence of account information on the electronic device camera is then checked, and if none are found, it is determined that this is a first-time connection and information uniquely identifying the electronic device is sent to the website server so that the server can set-up the account information based on the electronic device information. The server then sends user account information to the device, including an account ID and password. The user account information is then stored on electronic device for use the next time the electronic device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network.

22 Claims, 6 Drawing Sheets



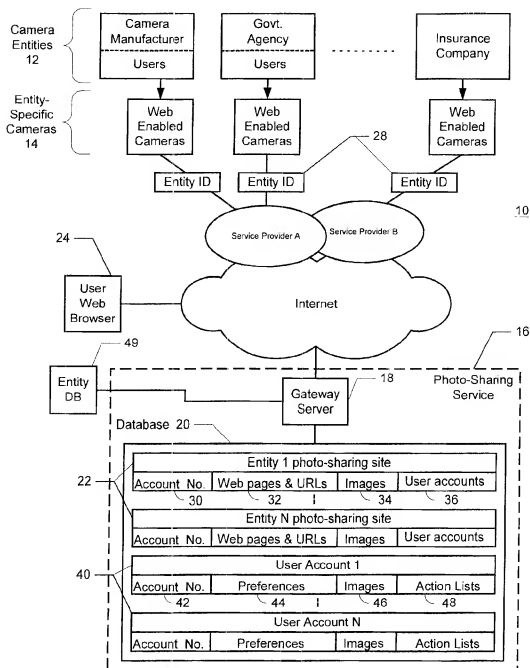


FIG. 1

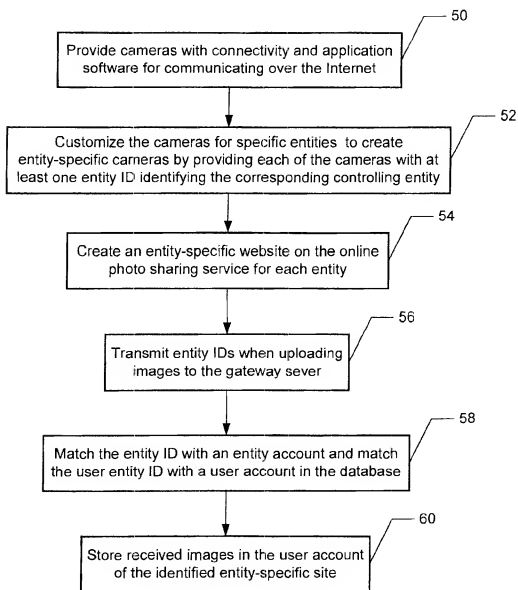


FIG. 2

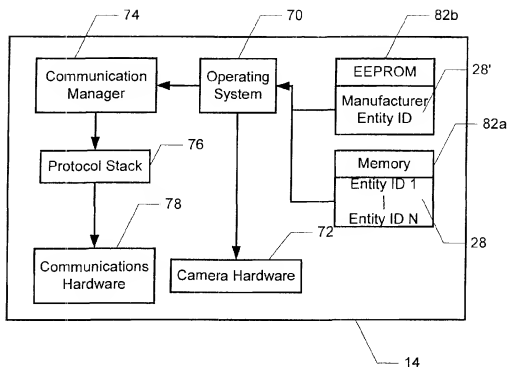


FIG. 3

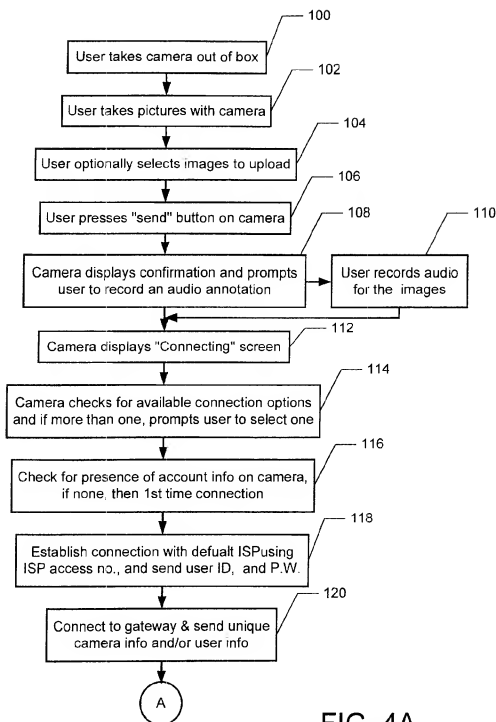


FIG. 4A

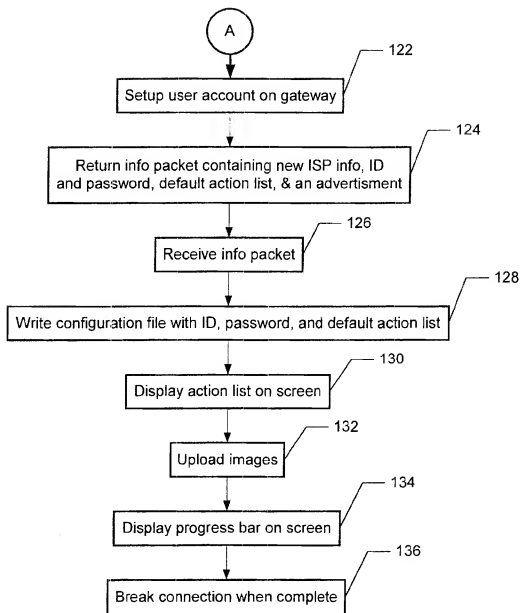


FIG. 4B

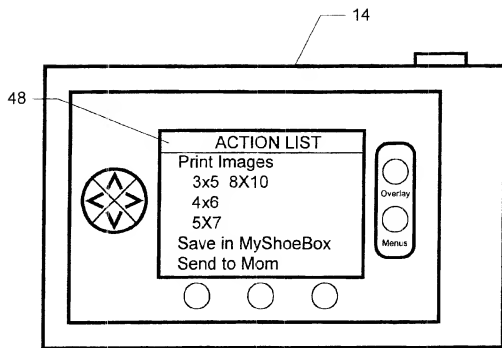


FIG. 5

1

AUTOMATICALLY CONFIGURING A WEB-ENABLED DIGITAL CAMERA TO ACCESS THE INTERNET

CROSS-REFERENCE TO RELATED APPLICATIONS

The present invention is related to co-pending U.S. patent application Ser. No. 09/625,398 entitled "Method and System For Hosting Entity-Specific Photo-Sharing Websites For Entity-Specific Digital Cameras"; and to co-pending U.S. patent application Ser. No. 09/626,418, entitled "Method And System For Selecting Actions To Be Taken By A Server When Uploading Images," which are assigned to the assignee of the present application and filed on the same date as the present application.

FIELD OF THE INVENTION

The present invention relates to a method and system for customizing digital cameras to upload images to an entity-specific photo-sharing websites, and more particularly to automatically configuring a web-enabled digital camera to access the Internet.

BACKGROUND

As the popularity of digital cameras grows, the desire of digital camera users to share their images with others will also continue to grow. New digital camera owners typically try to share their images based on the paradigm of film cameras, in which images are printed on paper and then placed into a photo album. The most straightforward approach to do this with a digital camera is to connect the digital camera directly to a printer to create the prints, and then manually insert the images into a photo album. Users often find this process somewhat complicated and restrictive because standard printers can only print images in limited sizes and requires particular types of paper. And even after the photo album has been assembled, the printed images are not easily shared with many people.

The best approaches to photo-sharing take advantage of the Internet. One such approach is for users to store the digital images on a PC and then send the images to others using email. Several Internet companies now offer an even more convenient approach by providing photo-sharing websites that allow users to store their images for free and to arrange the images into web-based photo albums. Once posted on a photo-sharing website, others may view the images over the Internet.

While convenient for storing digital images, getting the images to the photo-sharing websites can be challenging for users. Most commonly, users must upload their images from the digital camera to a PC using a cable or IrDA, or by inserting the camera's flash card into the PC. From the PC, the user logs onto the Internet and uploads the images to a photo-sharing website. After uploading the images, the user works on the website to arrange the images into web albums and to add any textual information.

Although today's approach for storing images from a digital camera onto a web photo-sharing website and for creating web photo albums works reasonably well, two problems remain that hinder the mainstream adoption of web-based photo-sharing. One problem is that this approach requires the use of a PC, notebook computer, or PDA. While many digital camera users today have PC's, most of those users are early adopters of technology. There are many other consumers who would purchase a digital camera, but are

2

reluctant to do so because they do not yet own a PC or are intimidated by them.

In an effort to address this problem, the assignee of the present application developed an approach to uploading images to the web that doesn't require the use of a PC. In this approach, an email software application is loaded into a digital camera capable of running software that allows the user to e-mail the images directly from the camera. The user simply connects his or her digital camera to a cellphone or modem, runs the e-mail application, and selects the desired images and the email recipients. The selected images are then sent to the recipients as e-mail attachments.

Although emailing photos directly from the camera allows users who do not own a PC to share images over the Internet, these users must still establish accounts with both an Internet service provider (ISP) and the photo-sharing website before being able to post their images. These accounts must also be set-up by PC users as well. For techno savvy users who use a PC to upload the images to the photo-sharing website, establishing the accounts may not be a bother, but even these users may not always have their PC's handy, such as when on vacation, for instance. And for non-PC users, establishing the accounts by entering account information on the digital camera itself may prove to be a cumbersome, if not a daunting, task.

Accordingly, what is needed is an improved method for uploading images from a digital camera to a photo-sharing website on the Internet. In order for online photo-sharing to become more mainstream, an approach that doesn't require a PC or PC expertise and that reduces complexity for the user is required. The present invention addresses such a need.

SUMMARY OF THE INVENTION

The present invention provides a method and system for automatically establishing a user account and for configuring a hand-held electronic device for accessing a site on a public network using the user account. The method and system include establishing a connection to a website server, which is responsible for establishing and maintaining website accounts. The presence of account information on the electronic device camera is then checked, and if none are found, it is determined that this is a first-time connection and information uniquely identifying the electronic device is sent to the website server so that the server can set-up the account information based on the electronic device information. The server then sends user account information to the device, including an account ID and password. The user account information is then stored on electronic device for use the next time the electronic device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network.

According to the method and system disclosed herein, a user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network with the electronic device.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG 1 is a block diagram illustrating an online photo-sharing system in accordance with a preferred embodiment of the present invention.

FIG 2 is a flow chart illustrating a business method for hosting entity-specific photo-sharing websites for entity-specific image capture devices in accordance with a preferred embodiment of the present invention.

3

FIG. 3 is a diagram showing a preferred embodiment of the connectivity and application software of the camera.

FIGS. 4A and 4B illustrate a flow chart of a process for automatically configuring a web-enabled digital camera to access the Internet in a preferred embodiment of the present invention.

FIG. 5 is a block diagram illustrating an example action list 48 displayed on the LCD screen of the camera.

DETAILED DESCRIPTION

The present invention relates to an automatic system for uploading images from a digital camera to entity-specific photo-sharing websites and for automatically establishing accounts. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

FIG. 1 is a block diagram illustrating an online photo-sharing system 10 in accordance with a preferred embodiment of the present invention. The system includes a plurality of camera controlling entities 12 that produce, own, or otherwise control a set of digital cameras 14, and an online photo-sharing service 16. The online photo-sharing service 16 includes a gateway server 18 and an entity/account database 20. The various camera controlling entities 12 contract with the photo-sharing service 16 to transparently host customized photo-sharing websites 22 for each entity, which are referred to herein as entity-specific photo-sharing websites 22. The entity-specific photo-sharing websites 22 are each accessible to the user through the entity's existing Internet site (not shown), and thus appear to users as though the entity-specific photo-sharing websites 22 are hosted by the corresponding entity. According to a preferred embodiment of the present invention, the cameras 14 for a particular entity are customized for that entity to create entity-specific cameras 14, such that when the cameras 14 connect to the Internet, the cameras 14 automatically upload their images to the photo-sharing website of the corresponding entity. In a further aspect of the present invention, the photo-sharing service 16 automatically stores the images in a web album, which is viewable over the Internet by a user's web browser 24.

As used herein, a camera controlling entity 12 is any entity that makes, owns, sells, or controls digital cameras 14, and therefore includes, camera manufacturers, companies, retailers, and end-users. One or more combination of these entities 12 may either contract with the photo-sharing service 16 to provide entity-specific websites 22 for their cameras 14, or have entity information transmitted to the photo-sharing service 16 from the cameras 14. Therefore, a camera controlling entity 12 may include a single entity 12 or a hierarchical relationship of entities 12.

An example of a single entity 12 includes an insurance company that contracts with the photo-sharing service 16 to have all digital cameras 14 used by their agents to transmit their images to a customized insurance photo-sharing website. Examples of a hierarchical relationships of entities 12 includes a camera manufacturer, such as Nikon, that contracts with the photo-sharing service 16 to have all Nikon digital cameras 14 transmit their images to the customized

4

Nikon photo-sharing website. Since the images of different users must be distinguished, each user of a Nikon camera 14 would also constitute an entity within the Nikon website so that the images from different users can be distinguished. Other examples of hierarchical entity relationships include a retailer and its consumers, a real estate agency and its agents, community groups and its members, and government agencies and its employees, for instance.

In a preferred embodiment, the cameras 14 are customized for their respective entities 12 by providing the cameras 14 with software for transmitting entity ID information 28 identifying its controlling entity 12 to the photo-sharing service 16. The photo-sharing service 16 in conjunction with the gateway server 18 and the entity/account database 20 hosts the entity-specific photo-sharing websites 22. Each entity-specific website 22 is identified in the database 20 by an entity account number 30 and includes the web pages and URIs 32 comprising the website, the images and web albums 34 stored on the website, and the user account numbers 36 of authorized users. The database 20 also includes user accounts 40, each of which comprises a user account number 42, user preferences 44, the user's images 46, and action lists 48, explained further below.

The gateway server 18, which communicates with the cameras 14 during image uploading, receives one or more entity IDs 28 from each camera 14 and matches the entity ID 28 with an entity account 30 in the database 20. The images are then automatically associated with the photo-sharing website 22 of the identified entity 12 and/or the identified user.

After the images are uploaded, a user of the camera 14 may visit the online photo-sharing website 22 over the Internet to view the images via a web browser 24. Since the photo-sharing websites 22 are transparently hosted by the photo-sharing service 16, each photo-sharing website 22 appears as though it is hosted by the entity itself, rather than the third party service.

In one embodiment, the cameras 14 may connect to the Internet via a service provider 26, which may include a wireless carrier and/or an Internet service provider (ISP) that is capable of servicing many devices simultaneously. In a preferred embodiment, each of the cameras 14 is provided with wireless connectivity for connecting to the Internet, and are therefore so called "web-enabled" devices, although a wired connection method may also be used.

The cameras 14 may be provided with wireless connectivity using anyone of a variety of methods. For example, a cellphone may be used to provide the digital camera 14 with wireless capability, where the camera 14 is connected to the cellphone via a cable or some short-range wireless communication, such as Bluetooth. Alternatively, the camera 14 could be provided with built-in cellphone-like wireless communication. In an alternative embodiment, the digital camera 14 is not wireless, but instead uses a modem for Internet connectivity. The modem could be external or internal. If external, the camera 14 could be coupled to modem via any of several communications means (e.g., USB, IEEE1394, infrared link, etc.). An internal modem could be implemented directly within the electronics of camera 14 (e.g., via a modem ASIC), or alternatively, as a software-only modem executing on a processor within camera. As such, it should be appreciated that, at the hardware connectivity level, the Internet connection can take several forms. Hence, it should be appreciated that the present invention is not limited to any particular method of accessing the Internet.

5

In a preferred embodiment, the entity-specific websites 22 are customized to seamlessly integrate into the entity's existing website by following the look and feel of the entity's existing website. The entity-specific websites 22 are hosted on the photo-sharing service 16, but a link to the entity-specific websites 22 may be provided on the homepage of the corresponding entity's existing website. Thus, in order to view a web album on an entity-specific website 22, the user must visit the entity's existing website and click the link to the entity-specific website 22, where the user's browser 24 will be transparently directed to the photo-sharing service 16 and be provided with the web pages 32 of the entity-specific website 22.

As an example of the operation of the photo sharing system 10, consider the following scenario. Assume that Minolta and Nikon are entities 12 that have contracted with the photo-sharing service 16, and that the photo-sharing service 16 hosts a photo-sharing website 22 for Minolta and a photo-sharing website 22 Nikon. The Minolta cameras 14 would be provided the entity ID 28 for Minolta and the Nikon cameras 14 would be provided the entity ID 28 for Nikon. When the Minolta and the Nikon cameras 14 send sets of images to the photo-sharing service 16, the gateway server 18 would distinguish the cameras 14 by the entity IDs 28 and would direct the set of images received from Minolta cameras 14 to Minolta's photo-sharing website, and would direct the images from Nikon cameras 14 to Nikon's photo-sharing website. To view the images, the owners of the cameras 14 would use a browser 24 on their PC or PDA to visit the URL of the Minolta or Nikon photo-sharing websites 22. In one preferred embodiment, the photo-sharing service 16 sends the URL of the entity-specific website 22 directly to the camera 14 for display to inform the user of the address.

According to the present invention, the photo-sharing service 16 provides business-to-business and business-to-consumer business models. The service is business-to-business because the service provides companies, such as camera manufacturers, with a complete end-to-end solution for their cameras 14. The solution includes customized software for their cameras 14 for sending images over the internet, and an internet website for storing images from those cameras 14 on a branded website that appears to be hosted by the company. The service is business-to-consumer because it allows users of digital cameras 14 with an automatic solution for uploading captured images from a digital camera 14 to an online photo-sharing website, without a PC.

According to one preferred embodiment of the present invention, the photo-sharing service 16 provides a method of doing business whereby the photo-sharing service 16 shares revenue based on the hierarchical relationship of the entities 12. For example, if the photo sharing service 16 charges a fee for receiving and/or storing the images received from the entity-specific cameras 14, then the photo sharing service 16 may share a portion of the fee with the manufacturer and/or third party supplier of the camera 14 that uploaded the images, for instance. Revenue may also be shared with the wireless service provider providing the connection with the photo sharing service 16.

FIG. 2 is a flow chart illustrating a business method for hosting entity-specific photo-sharing websites for entity-specific image capture devices, such as digital cameras, in accordance with a preferred embodiment of the present invention. First, the cameras 14 are provided with connectivity and application software for communicating over the Internet in step 50. In a preferred embodiment, this step is

6

performed during camera 14 manufacturing to provide off-the-shelf web enabled cameras 14. The cameras 14 are also customized for specific entities 12 to create entity-specific cameras 14 by providing each of the cameras 14 with at least one entity ID 28 identifying the corresponding controlling entity in step 52.

Referring now to FIG. 3, a diagram showing the preferred embodiment of the connectivity and application software of the camera 14 and the entity ID 28 information is shown. Preferably, the camera 14 includes a microprocessor-based architecture that runs an operating system 70 for controlling camera hardware 72 and overall functionality of the camera 14 (e.g., taking pictures, storing pictures, and the like). An example of such an operating system 70 is the Digita™ Operating Environment developed by the assignee of the present application. The camera 14 also includes communication manager 74 software, and a TCP/IP protocol stack 76, that enables communication via the internet, as is well-known in the art. The entity ID information 28 and captured images may be stored in one or more types of memories 82.

For hierarchical entity relationships, the cameras 14 are provided with hierarchical entity IDs 28; one entity ID 28 identifying the entity, and a second entity ID 28 identifying the end-user. Whether there are one or more entity IDs 28, the entity ID 28 of the camera manufacturer may always be provided. Camera 14 customization may occur either during manufacture or anytime thereafter. In a preferred embodiment, the manufacturer entity ID 28 is provided at the time of manufacturing and is stored in an EEPROM 82b, while the entity IDs 28 for other entities 12, such as companies and end-users, are loaded into the camera 14 subsequent to manufacturing and are stored in flash memory 82a or the EEPROM 82b.

Customization that occurs subsequent to manufacture may be implemented using several methods. The first method is to manufacture the cameras 14 with an application programming interface (not shown) for accepting a subsequently loaded software application that specifies the entity ID's 28. The application may come preloaded on a flash card, which is then inserted into the camera 14 by the user and stored in flash memory 82a. The application may also be wirelessly beamed into the camera. When executed in the camera, the software application transmits the appropriate entity IDs 28 to the gateway server 18.

The second method is to load a small file in the camera 14 specifying the entity IDs 28 from a removable memory or from a PC, and storing the file in a system folder within the camera's flash memory 82a. The camera 14 then accesses the file when an Internet connection is established. In a preferred embodiment, the communication manager 74 automatically extracts the manufacturing ID 28 and the entity ID 28 and transmits them to the gateway server 18. In this embodiment, the entity ID 28 is also stored in the EEPROM 82b and is factory set to zero (empty). Thus, unless the entity ID 28 is set, the manufacturing ID 28 may default as the highest controlling entity.

If, for example, a third party developer X contracts to provide custom camera software for camera manufacturer Z, then a custom entity ID will be issued for developer X and developer X will place the custom entity ID into the EEPROM 82b. Developer X is now a controlling entity 12, and may specify to the photo-sharing service 16 that a developer X entity-specific photo-sharing site 22 or developer X's own website be the destination for the uploaded images.

7

The protocol stack 76, under direction of the communications manager interfaces with the communications hardware 78 of camera. The protocol stack 76 includes software APIs and protocol libraries that interface with the communication manager 74, and communication hardware interface drivers that interfaces directly with the various communication hardware 72 the camera 14 must function with (e.g., a Bluetooth transceiver, etc.). The communication manager 74 communicates with operating system 70 and the IP protocol stack 76 to establish an Internet connection ant to transmit the entity ID 28 information and images from the memories 82a and 82b to the photo-sharing service 16.

In an alternative embodiment, rather than loading entity ID's 28 into the camera, a combination of the camera's serial number and the make and model number of the camera may be used as the entity ID 28. Entity specific cameras 14 may then be distinguished by providing a mapping of the camera serial numbers and product IDs to specific entities 12 in the database 20.

Although the camera 14 has been described in terms of a software-based customization solution, those with ordinary skill in the art will readily recognize that the camera 14 may also be provided with a hardware-based solution.

Referring again to FIG. 2, before or after camera customization, the entity-specific websites 22 are created for each entity contracting with the photo-sharing service 16 in step 54. Customization requires storage of entity information in the entity/account database 20 and creating and storing web page elements comprising the entity-specific photo-sharing website in the database 20. The entity-specific information stored in the database 20 may also include service levels, and enabled features for the entity-specific website 22. Features are components or services that may be provided on websites by the photo-sharing service 16, such as search functions, and online printing, for instance, but may be selectable by each entity for its own website. As an example, company X may provide customized cameras 14 for its employees, but may not wish to allow employees to print images from the company X photo website for security reasons. If so desired, company X may have the photo service disable this feature from their particular website.

In a preferred embodiment, the entity-specific websites 22 are not created from scratch, but are created by modifying a preexisting template. The template may include several different sections, such as A, B, C and D, for instance. Assuming for example that the template used to create a website for Nikon, and section A is used to specify the name of the entity then the name Nikon would be inserted into that section. Other entity-specific content would be used to fill out the remaining sections. The Web pages comprising the Nikon specific photo-sharing website would then be provided with URL's unique to that website. The entity's regular website would be modified to include a link to the entity's photo-sharing website 22. In addition, the entity-specific photo-sharing website would include a link back to the entity's website. Entities 12 may have entity photo-sharing websites 22 created for them in one of two ways; automatically by logging into the photo-sharing service 16 and manually customizing the templates, or by having the entity photo-sharing website created for them.

Referring still to FIG. 2, when a camera 14 establishes an internet connection with the gateway server 18, the camera 14 transmits its entity IDs 28 and/or user entity ID 28 when uploading user selected images to the gateway server 18 in step 56. In response, the gateway server 18 matches the entity ID 28 with an entity account in the database 20 and

8

matches the user entity ID 28 with a user account 40 in the database 20 in step 58. The images received are then stored in the user account 40 of the identified entity-specific website 22 in step 60.

Referring again to FIG. 1, each user account 40 in the database 20 may also include one or more action lists 48. According to the present invention, an action list 48 includes one or more items representing actions that the gateway server 18 should take with respect to uploaded images, such as where to store and/or send the images from a particular user or camera, for instance. As explained further below, the action list 48 stored on the database 20 under a user's account 40 are automatically downloaded to the user's camera 14 during a connection with the gateway server 18 and stored on the camera 14. When the user initiates an image upload, the action list 48 is displayed to the user so the user may easily select what actions the gateway server 18 should take with respect to the images by selecting the displayed action list items.

Examples of action list items include specifying that the uploaded images should be stored on the entity-specific photo website, sending the images to a list of email addresses, or even performing some type of analysis or calculation on the image data, for instance.

In a further aspect of the present invention, an action list item is not limited to, instructing the gateway server 18 to perform actions only within the photo-sharing service 16. Rather, an item in the action list 48 may also instruct the gateway server 18 to perform actions outside of the photo-sharing service 16, such as storing the images in an external database 49 of the entity 12. For instance, in the example where the entity 12 is a company, some users of the company's cameras 14 could have action lists 48 instructing the gateway server 18 to store uploaded images to the company's database, rather than to the company's photo-sharing site 22. Based on the action lists 48 and customization, the gateway server 18 may be programmed to automatically perform predefined tasks, such as creating new web albums, or a new page within an existing album, parse the images to extract sound files or other metadata, print images and mail them to designated addresses, and so on.

In a preferred embodiment, the action lists 48 may be created via several methods. In one method, the action list is created by the photo-sharing service 16 the first time the user's camera establishes a connection. That is, a default action list 48 is automatically created based on the entity ID when a user account 40 is first created. In a hierarchical entity relationship where the entity 12 is a company, a default action list 48 may be created to implement a workflow specified by the entity 12. In a hierarchical relationship where the entity 12 is camera manufacturer, for instance, a default action list 48 may be created instructing the gateway server 18 to store the user's images in a simulated "shoebox" on the entity-specific photo-sharing site 20. The user may then go online and create albums from the images in the shoebox as desired.

Another method is for the user to create the action list 48 online on the entity-specific photo-sharing site 20. The action list 48 may be created manually on the website 20 by the user navigating to the site 20 using a web browser 24, accessing her account, and manually creating the action list 48 or editing the action list 48 on the entity-specific site 22. The action list 48 may also be created automatically on the website 20 in response to user actions performed on the website, such as printing images, or creating a web album.

9

Alternatively, after performing an action, the user may be prompted whether they would like this action added to his or her action list 48. If so, the user clicks a check-box and the item is added the action list 48. In a preferred embodiment, any action list 48 created and edited on the photo-sharing site 20 are downloaded to the camera every time the camera 14 connects to the photo-sharing service 16 and made available for user selection on the camera 14 during the next upload.

Yet another method for creating an action list 48 is to allow the user to create the action list on the camera 14. The user may manually create an action list 48 by "typing" in predefined items on the camera. The user may also type in an email address as an action list item whereby when that item is selected, the uploaded images are stored as a web album on the entity-specific photo-sharing website 22 and the server 18 sends a notification to the specified recipient containing the URL to the web album page.

A method and system for hosting web-based photo-sharing websites and for customizing digital cameras to upload images to the entity-specific photo-sharing websites has been disclosed. According to the present invention, users of the customized cameras 14 can upload images to the Internet for storage and web photo album creation without the use of a PC.

In one embodiment described above, the present invention assumes that an ISP account has been established with the digital camera's service provider, and that users of cameras 14 belonging to a certain entity 12 may use the cameras 14 to upload images to the website of the entity 12. However, two problems with account setup remain. One problem is that just as with a PC and PDA, the user must first establish an ISP account before the camera 14 can establish Internet communication. The second account problem is that most websites, including the photo sharing sites 22, may require each user to establish a unique account before using the site to distinguish one user from another. Before being able to connect the web-enabled camera 14 to the Internet, the user must establish these two accounts by either entering account setup information on a PC or entering account setup information on the camera 14. Neither alternative is a convenient alternative for people who do not have the time nor inclination to do so.

In a further aspect of the present invention, the cameras and the photo sharing site are provided with software for automatically creating Internet and photo-sharing website accounts for each camera 14 upon first use, without requiring the user to first enter account information on a PC or on the camera 14.

Referring now to FIGS. 4A and 4B, a flow chart illustrating a process for automatically configuring a web-enabled digital camera to access the Internet in a preferred embodiment of the present invention. Although the process will be described in terms of automatically establishing Internet accounts for a digital camera without requiring the user to enter information, those with ordinary skill in the art will readily recognize that the present invention may be used to automatically establish Internet accounts for any type of portable electronic device.

The process assumes that a user has just acquired a digital camera 14 customized as described above, and has just taken the camera 14 out of its box in step 100. After taking pictures with the camera in step 102, the user may review the images in the camera's LCD screen and optionally select a set of images to upload to the photo sharing service 16 in step 104. The user then presses a "send" button on the camera in step 106 to upload the images.

10

In response, the camera displays a confirmation dialog screen on the camera and prompts the user to record an audio annotation for the images or to continue in step 108. The user may then choose to record audio for the images in step 110. After choosing to continue, or after recording audio, the camera displays a "connecting" dialog screen in step 112 to indicate to the user that the camera is establishing an Internet connection. At the same time, the camera checks for available connection options in step 114, and if more than one is found, the camera prompts the user to select one of the connection options. For example, the camera may be within range of a Bluetooth-equipped printer and a cellphone, so the user will be prompted to choose which device the camera should establish communication with.

The camera then checks for the presence of account information on the camera in step 116, and if there are none, the camera assumes that this is a first-time connection. According to the present invention, in order to allow the camera to make a first-time Internet connection, the camera is provided with default Internet service provider (ISP) information during manufacturing, including an ISP access number, and user ID and password (if required). The camera establishes connection with the default ISP in step 118 by dialing the preloaded access number, and by sending the preloaded user ID and password to the ISP. This special account may be configured so that the camera can only connect to the gateway server 18 (no other IP addresses may be allowed).

After connecting with the ISP, the camera connects to the gateway server 18 and sends unique camera information and/or user information in step 120. In a preferred embodiment of the present invention, a combination of the camera's serial number and the make and model number of the camera may be sent as the unique camera information. In another preferred embodiment, the user's e-mail address may be sent as the unique camera or user information.

Continuing with FIG. 4B, the gateway server 18 uses the unique camera information to set up a user account 40 in step 122. After creating the user account 40, the gateway server 18 returns an information packet to the camera containing new ISP information (if needed), an account ID, and an account password in step 124. The information packet may also contain a default action list specifying what actions should be taken with respect to the images, an advertisement for display on the camera, and the URL of the entity-specific website 22.

It should be noted that if the camera is used in conjunction with an IP direct phone or is provided with a phone number for connected to a dedicated server where the user is not billed separately for the ISP connection, then the steps of providing the camera with default ISP info and returning new ISP info, may be omitted.

The camera receives the information packet in step 126, and writes a configuration file to memory 82a containing the ID, password, and default action list in step 128. The camera then displays the action list on the camera's LCD screen for selection by the user in step 130. The camera may optionally display the user's account information as well.

In an alternative preferred embodiment where security is a concern, the camera 14 first logs off the special ISP and the gateway accounts, and then reconnects using new ISP and gateway accounts in order to retrieve the action lists 48.

FIG. 5 is a block diagram illustrating an example action list 48 displayed on the LCD screen of the camera 14. The action list 48 is shown displaying three major options; printing the uploaded images, saving the uploaded images in

11

the user's shoebox, and sending the images to Mom. Under the printing option, the user may select from various size prints. Rather than nested menu categories as shown under the printing option, the action list 48 may be displayed with each action listed as a separate item (e.g., "Send 4x8 prints to Mom", "Send 5x7 prints to me").

Referring again to FIG. 4B, after the user selects one or more actions from the action list 48, the camera begins to upload the images along with the selected actions in step 132 and displays a progress bar on the screen in step 134. In one preferred embodiment, the camera may also display the advertisement sent in the information packet from the gateway server 18. The advertisement may advertise the controlling entity 12, the entity's photo-sharing site 22, or the photo sharing service 16. After all the images are uploaded and associated with the user's account 40, the camera breaks the connection with the gateway server in step 136. At this point, the camera 14 may also display the URL of the entity-specific website 22 to the user.

The next time the user uploads images, the camera will use any new ISP information received to connect to the Internet and will use the account ID and password written to memory 82a when connecting to the gateway server 18. Thus, by using unique camera information, such as the serial number, to establish a web site account upon first use, the present invention eliminates the need for the user to type in information to establish a web site accounts.

To further explain the present invention from a user interaction point of view, consider the following scenario where a user named Jack has just purchased a digital camera 14 from a store and unpacks the camera from the box. There is a "Quick Start Guide" which guides him in getting started. Jack pops in the batteries, sets the date and time, and takes some pictures of his dog and his new baby. Jack can see the pictures have come out well on the small LCD, and now wants to try sharing them with his parents.

The Quick Start Guide says to select the photos to be sent using the "Select" button, and then press the "Send" button. So, Jack navigates to each of the baby pictures, selects them, and then presses the "Send" button. Instantly, a dialog comes up on the LCD screen: "No Receiving Device Found! Please turn on your phone or other connecting device." Oops! Jack pulls out his cell phone, and turns it on. He presses the "Continue" button. Jack does not see this happen, but the camera now "discovers" the cell phone, and immediately presents another dialog: "4 Images Selected. Press Record to add a Sound Note, or Continue to send". Although Jack finds the proposition of recording sound intriguing, he decides to skip it and presses the "Continue" button. Immediately, the dialog is replaced with a "Connecting . . ." dialog.

Shortly, another dialog appears: "Your camera serial number is 38147. Please write this down. You will need it to access your web photo albums". Jack writes the number down on the spot provided in the Quick Start Guide, and presses OK.

Another "Connecting . . ." dialog appears, and is then quickly replaced by another dialog, which says "A free, temporary account has been set up for you at www.photo-sharing.service.com/new_accounts. You will need your camera serial number to access your photos and complete the setup of your account. Please complete the account setup within 30 days". Jack writes down the URL in the space provided in the Quick Start Guide, and presses the OK button. Jack doesn't know it, but during this dialog, the camera has begun transmitting his images and is already partially complete.

12

A new dialog comes up, with a progress bar. Jack is surprised to see that the transmission is already almost ½ done. Below the progress bar, the dialog says "Press 'Continue' to use camera during photo transmission, or wait for progress bar to complete". Jack is interested in watching how fast his images are transmitted, and decides to watch the progress bar complete. A "Transmission Successful" dialog appears. Jack presses the OK button. The camera returns to the review mode.

Jack is pretty excited—he just sent four baby pictures to the Internet. Jack then decides to see what happened to his images so he turns on his PC. After connecting to the Internet, Jack types in the URL from the Quick Start Guide. A Photo-sharing service web page appears, welcoming Jack to the photo-sharing service. After looking briefly at the welcome page, Jack types in his serial number from the Quick Start Guide, and selects his camera's model number from a pop-up menu. Jack clicks on a "Submit" button on the web page.

Jack now sees a page which shows thumbnails of the baby pictures he just sent, the page is entitled "My Shoebox". The page explains to Jack that he is looking at his on-line digital photo shoebox. Since the server 18 knows this is Jack's first visit, special help messages may appear. Various options are provided via buttons and text links. One that catches Jack's eye is CREATE WEB PHOTO ALBUM. Jack clicks this button, and works his way through the process of setting up an on-line photo album. This includes selecting photos from the shoebox, as well as selecting layout and style. One of the check-box items Jack is offered is "Make this album a camera Action List Item". Jack doesn't know what that is, so he clicks the Action List link, which brings up a brief description "If you check this box, you will be able to send pictures directly from the camera to this photo album!" Jack finds this interesting, so he closes the description window, and checks the box. Jack also enters the email address for his parents and his wife's parents, so they can be notified to come and see his photo album, which he has entitled "Our First Baby".

One of the buttons Jack does not click is the "Complete Account Setup" button. He knows that he has 30 days to do that chore, and figures he will get back to it later.

Jack's wife arrives home from shopping at this point, and Jack wants to show her how the new camera works. He starts by showing her the baby album on the PC, then decides to take some pictures of them holding the baby. Jack then selects the images, and presses the "Send" button again.

Since his cell phone is still on, sitting on the table a few feet away, the connection goes smoothly and quickly. A new dialog pops up, surprising Jack. It says "Select the destination for your pictures" and offers two choices: "My Shoebox" and "Our First Baby". Jack is amazed, he doesn't realize that the server 18 downloaded his action list to his camera during the connection. Jack decides to select the "Our First Baby" web album as the destination for the images, and clicks OK. The pictures are sent as before. After the transmission has completed, Jack goes to the PC to check on the photo album. When Jack refreshes the album page, he now sees the additional pictures he just sent, along with the pictures he sent before.

Jack spots a "Send Prints" link on the web page, and clicks it. He is led through a selection of print types, mailing addresses, and credit card info to make it possible to send prints. He is offered the option of completing the setup of his account. Jack decides to do that now, and proceeds to fill out the requested information, including his credit card number.

13

Once the account setup is complete, Jack continues the print order. One of the radio button items is "Make prints a separate Action List Item" or "Make prints part of your Action List". Jack remembers something about Action Lists from before, but is not sure what this means. The description says "Making a separate action list for prints allows you to decide in the camera to send to the photo album and send prints or to just send to the photo album." Jack thinks this is cool, and clicks the "Make prints a separate Action List" item. The next time Jack sends photos from the camera, his action list will be updated to present three choices: My Shoebox, Our First Baby, and Our First Baby w/Prints.

The underlying technology supporting this scenario are summarized below. Other functions and features are assumed, but not required for this scenario:

1. Two-way connection between camera and portal
2. Camera metadata included in the request
3. If not using an IP direct connection,
 - 3.1. Default ISP connection info built into the camera for the first connection (country specific)
 - 3.2. Downloaded assigned ISP information from the portal to the camera
4. Software capable of recognizing automatically a set of supported phones and adjusting the protocol to match
5. Action Lists maintained on the server that are automatically downloaded to the camera to update the camera selection list each time a connection is made
6. An On-line shoebox
7. Ability to download files to the camera from the server. The files could be text, GIF, animated GIF, JPG, or even a script or applet. This feature enables the ability to display advertisements on the camera, remind the user of remaining time to complete account setup, make special offers, and indicate limits reached.

A method and system for automatically configuring a web-enabled digital camera to access the Internet has been disclosed. Although the present invention has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations to the embodiments and those variations would be within the spirit and scope of the present invention. For example, although the photo-sharing service has been described as including the gateway server and the database, the database may be located elsewhere. Also, the gateway server may be used to control account information, while one or more other servers may be used to provide the web pages of the entity-specific websites. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.

What is claimed is:

1. A method for automatically establishing a user account and for configuring a hand-held image capture device which includes a digital camera for accessing a site on a public network using the user account, the method comprising the steps of:

- (a) establishing a connection to a website server, the website server being responsible for establishing and maintaining website accounts;
- (b) checking for the presence of account information on the image Capture Device and if none are found, determining that this is a first-time connection and automatically sending image capture device information uniquely identifying the electronic device to the website server so that the server can set-up the account information based on the image capture device information;

14

(c) receiving user account information from the server, wherein the user account information includes an account ID and password; and

(d) storing the user account information on image capture device for use a next time the image capture device accesses the website, whereby the user does not have to enter account information in order to establish the connection with the public network or to establish the website account.

2. The method of claim 1 further including the step of providing a digital camera as the hand-held image capture device.

3. The method of claim 2 further including the step of sending at least a partial serial number as the image capture device information uniquely identifying the electronic device.

4. The method of claim 3 further including the step of uploading captured images to the website server.

5. The method of claim 4 wherein step (a) further including the steps of:

(i) providing the image capture device with default ISP information; and

(ii) establishing a connection to an ISP using the default ISP information.

6. The method of claim 5 further including the step of terminating the connection with the ISP when the uploading of the images is complete.

7. The method of claim 6 further including the step of providing the digital camera with a default ISP access number, user ID, and password.

8. A system for automatically establishing a user account and for configuring a hand-held image capture device which includes a digital camera image capture device for accessing a site on a public network using the user account, comprising:

means for establishing a connection to a website server, the website server being responsible for establishing and maintaining website accounts;

means for checking for the presence of account information on the image capture device, and if none are found, determining that this is a first-time connection and automatically sending image capture device information uniquely identifying the image capture device to the website server so that the server can set-up the account information based on the image capture device information; and

means for receiving user account information from the server, wherein the user account information includes an account ID; and

means for storing the user account information on image capture device for use a next time the image capture device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account.

9. The system of claim 8 wherein the hand-held image capture device comprises a digital camera.

10. The system of claim 9 wherein the image capture device information uniquely identifying the image capture device includes a serial number of the image capture device.

11. The system of claim 10 wherein the camera uploads captured images to the website server.

12. The system of claim 9 wherein the camera further includes means for providing the image capture device with default ISP information, and means for establishing a connection to an ISP using the default ISP information.

13. The system of claim 12 wherein the digital camera terminates the connection with the ISP upon completion of image uploading.

15

14. The system of claim 13 wherein the camera is provided with a default ISP access number, user ID, and password.

15. A method for automatically establishing a user account and for configuring a digital camera to access a website on a public network using the user account, the digital camera including captured images and communication means for accessing the public network, the method comprising the steps of:

- (e) storing default ISP information on the digital camera, the default ISP information including an access number;
- (f) in response to a user request to upload images to the website, determining whether this is a first-time connection;
- (g) if it is first-time connection, establishing communication with the default ISP by dialing the access number;
- (h) establishing communication with the website and automatically sending a digital camera ID to the website;
- (i) using the digital camera ID to set up a user account on the website;
- (j) sending user account information to the digital camera, the user account including an account ID and an account password;
- (k) storing the user account information on the camera; and
- (l) uploading images to the website and associating the images with the user account, whereby the user does not have to enter information to establish a connection with the public network or to establish the website account.

16. The method of claim 15 further including the step of sending at least a partial serial number as the digital camera ID.

17. The method of claim 16 further including the step of receiving new ISP information from the website.

18. The method of claim 17 further including the step of using the new ISP information a next time the digital camera establishes a connection with the public network.

16

19. A system for automatically establishing a user account and for configuring a digital camera for accessing a website on a public network using the user account, the digital camera including images captured by user and communication means for wirelessly accessing the public network, the system comprising the steps of:

- means for storing default ISP information on the digital camera, the default ISP information including an access number;
- means for determining whether this is a first-time connection in response to a user request to upload images to the website;
- means for the establishing communication with the default ISP by dialing the access number;
- means for establishing communication with the website and sending a digital camera ID to the website;
- means for using the digital camera ID to set up a user account on the website;
- means for automatically sending user account information to the digital camera, the user account including an account ID and an account password;
- means for storing the user account information on the camera; and
- means for uploading images to the website and associating the images with the user account, whereby the user does not have to enter information to establish a connection with the public network or to establish the website account.

20. The system of claim 19 wherein the digital camera ID includes a serial number.

21. The system of claim 20 wherein the digital camera receives new ISP information from the website.

22. The system of claim 21 wherein the digital camera uses the new ISP information a next time the digital camera establishes a connection with the public network.

* * * * *



US006784924B2

(12) **United States Patent**
Ward et al.

(10) **Patent No.:** US 6,784,924 B2
(45) **Date of Patent:** Aug. 31, 2004

(54) **NETWORK CONFIGURATION FILE FOR AUTOMATICALLY TRANSMITTING IMAGES FROM AN ELECTRONIC STILL CAMERA**

(75) Inventors: **Joseph Ward**, Hilton, NY (US);
Kenneth A. Parulski, Rochester, NY (US); **James D. Allen**, Rochester, NY (US)

(73) Assignee: **Eastman Kodak Company**, Rochester, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/004,046**

(22) Filed: **Jan. 7, 1998**

(65) **Prior Publication Data**

US 2003/0142215 A1 Jul. 31, 2003

Related U.S. Application Data

(60) Provisional application No. 60/037,963, filed on Feb. 20, 1997, and provisional application No. 60/037,962, filed on Feb. 20, 1997.

(51) Int. Cl.⁷ **H04N 5/225**

(52) U.S. CL. **348/207.1; 348/211.3**

(58) **Field of Search** 348/207, 211, 348/212, 213, 231, 232, 233, 552, 239, 211 99–211 13, 211.3, 17; 710/62, 3, 8; 345/327; 386/48, 117; 725/131–133, 139–141, 151–153, 709/217–219, 220, 705/26, 27, 455/566

(56) References Cited

U.S. PATENT DOCUMENTS

4,524,381 A 6/1985 Konishi
4,574,319 A 3/1986 Konishi
4,825,457 A 4/1989 Lebowitz
5,241,659 A 8/1993 Parulski et al.
5,477,264 A 12/1995 Sarbadhikari et al.

5,606,365 A 2/1997 Maurinus et al.
5,663,678 A 9/1997 Chang
5,666,159 A 9/1997 Parulski et al.
5,737,491 A * 4/1998 Allen et al. 704/270
5,800,005 A * 9/1998 Hull et al. 455/566
5,825,432 A * 10/1998 Yonezawa 348/563
6,111,604 A * 8/2000 Hashimoto et al. 348/220
6,122,005 A * 9/2000 Sasaki et al. 348/211.3
6,166,729 A * 12/2000 Acosta et al. 345/719
6,449,001 B1 * 9/2002 Levy et al. 348/1404

OTHER PUBLICATIONS

User's Manual—Axis 2420 Network Camera, 1996 (Axis Communications).*

U.S. patent application Ser. No. 60/037,963, Parulski et al. "The Universe of Smart Cards: EasySIM software", Schlumberger Limited.

"Claris Emailer" from Claris' home page on internet.

"Kodak Professional Digital Camera System—User's Manual," Eastman Kodak Company, 1991–1992, pp. x to xv, 2–1 to 2–3, 3–29 to 3–30, and 4–1 to 4–49.

* cited by examiner

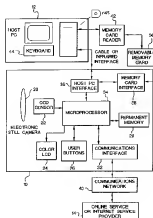
Primary Examiner—Tuan Ho

(74) Attorney, Agent, or Firm—Pamela R. Crocker

(57) ABSTRACT

A network configuration file is generated at a host computer and downloaded to a digital camera. This file contains instruction information for communicating with a selected destination via a communications interface. The digital camera includes a "send" button or LCD icon which allows the user to easily transmit one or more images via a wired or wireless communications interface to a desired destination, which among other possibilities may be an Internet Service Provider or a digital photofinishing center. When the user selects this option, the communications port settings, user account specifics, and destination connection commands are read from the network configuration file on the removable memory card. Examples of these settings include serial port baud rate, parity, and stop bits, as well as account name and password.

9 Claims, 4 Drawing Sheets



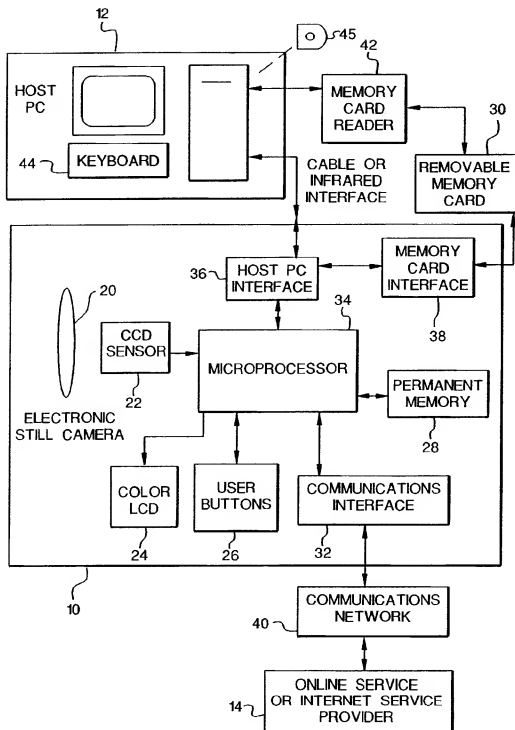
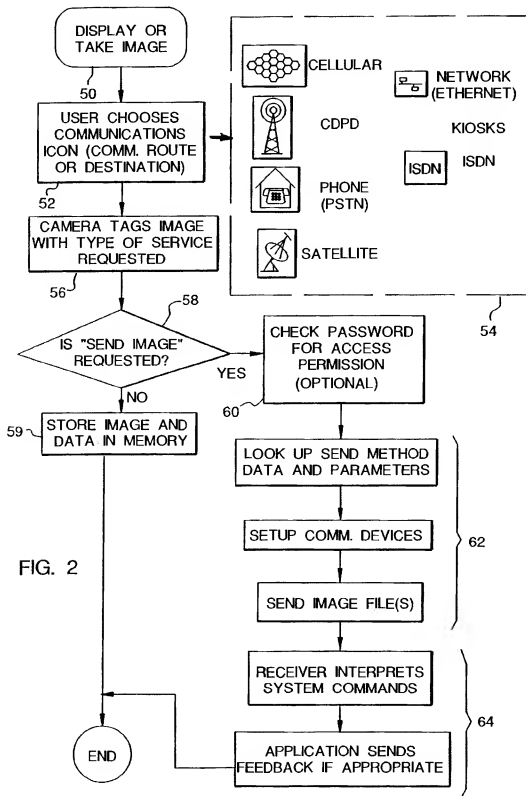


FIG. 1



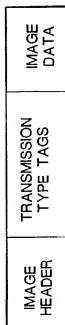


FIG. 3



FIG. 4

FIG. 4A

PHONE (PUBLIC SWITCHED TELEPHONE NETWORK)



PROTOCOL TYPE	PHONE NUMBER	DEFAULT SETTINGS	MODERN CONTROL STRING	ACCOUNT DATA	PASSWORD	SYSTEM COMMANDS
------------------	-----------------	---------------------	-----------------------------	-----------------	----------	--------------------

CELLULAR OR PCS

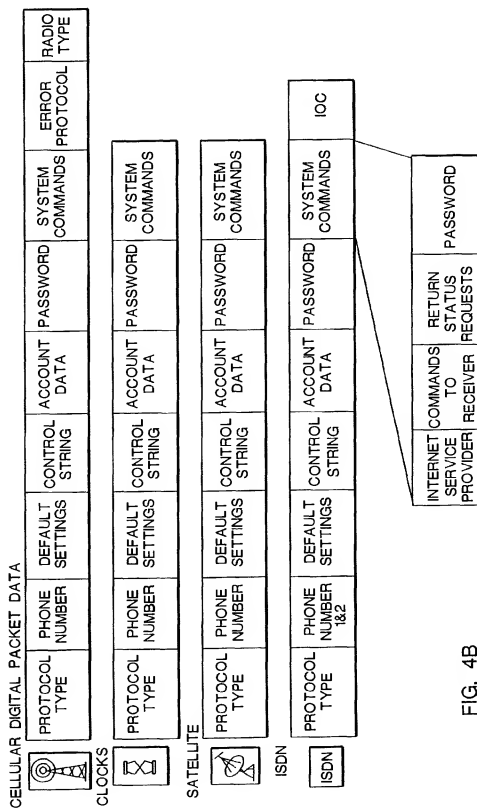


PROTOCOL TYPE	PHONE NUMBER	DEFAULT SETTINGS	MODERN CONTROL STRING	ACCOUNT DATA	PASSWORD	SYSTEM COMMANDS	ERROR PROTOCOL	RADIO TYPE
------------------	-----------------	---------------------	-----------------------------	-----------------	----------	--------------------	-------------------	---------------

WIRELESS LAN



PROTOCOL TYPE	PHONE NUMBER	DEFAULT SETTINGS	PARA- METER FILE	ACCOUNT DATA	PASSWORD	SYSTEM COMMANDS
------------------	-----------------	---------------------	------------------------	-----------------	----------	--------------------



1

NETWORK CONFIGURATION FILE FOR AUTOMATICALLY TRANSMITTING IMAGES FROM AN ELECTRONIC STILL CAMERA

CROSS-REFERENCE TO RELATED APPLICATION(S)

THIS APPLICATION CLAIMS BENEFIT OF PROVISIONAL APPLICATION SERIAL NO. 60/037,962 FILED Feb. 20, 1997.

Reference is made to commonly assigned copending applications Serial No. 60/037,963, filed Feb. 20, 1997 entitled "Electronic Camera with 'Utilization' Selection Capability" and filed in the names of Kenneth A. Parulski, Joseph Ward, and Michael C. Hopwood, which is assigned to the assignee of this application.

FIELD OF THE INVENTION

The invention relates generally to the field of photography, and in particular to electronic photography. More specifically, the invention relates to a digital camera that interfaces with a host computer.

BACKGROUND OF THE INVENTION

Digital cameras, such as the Kodak Digital Science DC25™ camera, allow images to be utilized on a home computer (PC) and to be incorporated into e-mail documents and personal home pages on the World Wide Web. Presently, images must be copied to the PC and transmitted as e-mail, for example using an online service or an Internet Service Provider (ISP), via a modem from the user's PC. It would be desirable to be able to transmit pictures directly from the digital camera instead of first transferring the pictures to a PC. For instance, on a vacation trip, it is desirable to immediately share pictures with friends or relatives via e-mail or Internet access. It is also desirable to transmit pictures from a location without PC access in order to free up camera storage to take additional pictures. There are a wide variety of connection means to online services such as America On Line, ISPs, and bulletin board services. Each of these services typically requires an account name and password, as well as local telephone access numbers, and specific communications settings. It would be difficult to provide an easy-to-use means with buttons or menus on a small digital camera to input and/or modify all of these required settings.

SUMMARY OF THE INVENTION

The present invention is directed to overcoming one or more of the problems set forth above. Briefly summarized, according to one aspect of the present invention, a network configuration file is generated at a host computer and downloaded to a digital camera. This file contains instruction information for communicating with a selected destination via a communications interface. The digital camera includes a "send" button or LCD icon which allows the user to easily transmit one or more images via a wired or wireless communications interface to a desired destination, which among other possibilities may be an Internet Service Provider or a digital photofinishing center. When the user selects this option, the communications port settings, user account specifics, and destination connection commands are read from the network configuration file. Examples of these settings include serial port baud rate, parity, and stop bits, as well as account name and password.

2

In addition, information about which image or images to transmit is entered using the user buttons on the digital camera. This information is used to automatically establish a connection, log-in to the desired destination, and to transmit the image. The transmission may occur immediately after the pictures are taken, for example if the camera has a built-in cellular phone modem, or at a later time, when the camera is connected to a separate unit (such as a dock, kiosk, PC, etc.) equipped with a modem. In the latter case, a "utilization file" is created to provide information on which images should be transmitted to which account.

These and other aspects, objects, features and advantages of the present invention will be more clearly understood and appreciated from a review of the following detailed description of the preferred embodiments and appended claims, and by reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a system block diagram of the invention.
FIG. 2 is a diagram showing the steps used to automatically transmit images using the network configuration file.
FIG. 3 is a diagram of an image file.
FIG. 4 is a diagram showing several versions of the network configuration file.

DETAILED DESCRIPTION OF THE INVENTION

Because imaging systems and devices are well known, the present description will be directed in particular to elements forming part of, or cooperating more directly with, apparatus in accordance with the present invention. Elements not specifically shown or described herein may be selected from those known in the art. Some aspects of the present description may be implemented in software. Unless otherwise specified, all software implementation is conventional and within the ordinary skill in the programming arts.

A system block diagram of the invention is shown in FIG. 1 including an electronic still camera 10, a host computer (PC) 12 and a service provider 14. The camera includes an optical section 20 for imaging a scene upon a CCD sensor 22 and generating an image signal, a liquid crystal display (LCD) 24 for displaying images and other information, a number of user input buttons 26, both permanent memory 28 and removable memory 30, and an internal communications interface 32 (e.g., modem). This interface may connect to a variety of known networks, such as a public switched telephone network (PSTN), ISDN, an RF cellular phone network, or Ethernet. The camera 10 also includes a microprocessor 34 for generally controlling the camera functions, as well as the interchange of data with the host PC 12 and the memory card 30 through a host PC interface 36 and a memory card interface 38, respectively. Besides the host PC 12, the system includes a network connection 40 to the online service or ISP (Internet Service Provider) 14. Alternately, the network 40 can connect to the user's home PC 12.

When the camera 10 is first purchased (or at any time thereafter), it is connected to the PC 12 via the host PC 36 interface and a software application (stored on a disc 45) running on the host PC 12 will enable the user to specify the name of a destination ISP or online service and to input from the host PC keyboard 44 the appropriate communication settings and account information. This information generates a network configuration file, which then can be downloaded to the camera 10 through the host PC interface

3

4

36, which may be a wired or infrared (e.g., IrDA) interface, and written to the camera's internal memory 28 and/or the removable memory card 30. Alternatively, a host PC equipped with a memory card reader/writer 42 can write the information directly to the card 30 without connecting the camera through its host PC interface 36. Also, this information could be predetermined by the user and stored in a "preferences" file on the host PC 12 and then transferred to the camera 10 from this file without further intervention by the user. Multiple sets of destination services can be stored on the memory card 30. Typically, keyword or graphic descriptors (e.g., icons) accompany the information in the network configuration file about destination services to enable easy access by the camera user.

The steps used to automatically transmit images using the network configuration file are shown in FIG. 2. After disconnecting the camera from the host PC, the user operates the camera to take pictures (step 50). This is typically done at a remote location, for example while traveling to another city. As the user takes or reviews images on the image LCD display, the decision can be made to transmit one or more images (step 52). This is done by choosing one of the keywords or icons in a menu 54 shown in FIG. 2, which are displayed on the LCD 24 and selected, e.g., through the user buttons 26. (Note that a camera will typically only include a subset (only those desired by the user) of all the different services shown.) The selected image files may be tagged with a code (step 56) indicating which service is requested, as shown in FIG. 3. (Alternately, an "image utilization" file can be created in the camera storing a list of images to be transmitted by a particular method, as described in the cross-referenced depending patent application (U.S. Serial No. 60/037,963). As described in that patent application, the details of an order, e.g., number of print copies to be made from an image and the size of the prints and/or a list of images to be e-mailed to various recipients, is written into the "utilization" file, which identifies the order and includes pointers to the image files that store the images required to "fulfill" the order. The "utilization" file is stored in the internal memory 28 or the memory card 30.)

Next, the system determines whether a request exists to send an image (step 58). If no request is present, the image and associated data is stored in either permanent memory 28 or the memory card 30 (step 59). (Typically, all images are initially saved in memory whether eventually sent or not.) Otherwise, if there is a request to send an image, the user ensures that the camera is connected to the appropriate service (wired telephone line, cellular phone, kiosk, etc.) and pushes a "send" button in the user button section 26, or selects a "send" menu option on the LCD 24. The camera then utilizes the appropriate network configuration file, shown in FIG. 4. Each network configuration file contains items such as the protocol type, phone number, etc., as described in Appendix I. The user password may be checked against the password in the network configuration file to ensure that the user is authorized to connect the camera to the desired service (step 60). Alternately, the stored password in the appropriate configuration file can be used. Next, the camera uses the parameters in the configuration file to establish communications with the service and send one or more image files as selected by the user (steps 62). The service receiver interprets the system commands issued by the camera from the network configuration file list and sends appropriate feedback (such as "transfer in progress" and "transfer complete") which are interpreted by the camera and displayed on the LCD 24 (steps 64).

For example, when the camera uses a normal wired telephone (Public Switched Telephone Network) connection

(i.e., network 40) to the camera's internal modem 32, after the user selects the images to be sent and presses the "send" button, the camera performs the following steps without user intervention:

- 1) Read the appropriate connection parameters from the network configuration file (on the memory card 30 or internal camera memory 28), dial the phone and establish the connection to the destination service 14.
- 2) Read the user's account name and password and transmit these to "log-on" to the service 14.
- 3) Using the appropriate communications protocol (FTP, mailto, etc.), transmit the selected image or images to the destination service 14.

The invention has been described with reference to a preferred embodiment. However, it will be appreciated that variations and modifications can be effected by a person of ordinary skill in the art without departing from the scope of the invention.

Appendix I

These are descriptions of the tags listed in the previous drawing:

Protocol Type

Each communication method has its own protocol, or rules to communicate. This tag identifies that protocol and where to find it. For example, the Network may use TCP/IP and a modem may use XModem.

Phone Number

This is the number of the receiving service. If internet access is requested, this could be the number of the Internet Service Provider. For ISDN, some systems require two phone numbers, dialed and connected to in sequence.

Default Settings

Standard settings that make the communications device compatible with the imaging device.

Modem Control String

Modem and communications devices have a command language that can set them up before they are used. For example, modems have many options controlled by command strings including volume level, the amount of time the carrier is allowed to fail before the system hangs up, and so on.

Account Data

This can be internet account data, charge number data, phone card data, billing address, and data related to the commerce part of the transmission.

Password

Any password needed to get into the communications system. Other passwords to get into the remote application or destination are located in the System Commands section.

System Commands

These are commands that control the end destination.

Error Protocol

In cellular and some other wireless communications, error protocols are used to increase the robustness of the link. For example, MNP10 or ETC may be used for cellular links.

Radio Type

The type of radio used for this communications feature may be identified here. Some cell phones have modems built in, others will have protocols for many communications functions built in. The radio type will make the imaging device adapt to the correct interface.

IOC

ISDN Ordering Code identifies what features are available on the ISDN line provided by the telco. It is used to establish the feature set for that communications link.

5

Internet Service Provider

This identifies the actual service provider and any specific information or sequence of information that the service wants to see during connection and logoff. It also tells the device how to handle the return messages, like "time used" that are returned by the server.

Commands to Receiver

This may be a list of commands to control the receiving application. For example, a command to print one of the images and save the data to a particular file on a PC may be embedded here.

Return Status Requests

This tag can set up the ability of the application to tell if an error has occurred, or what the status of the application might be. The data here will help the device decide if it should continue communicating and a set user interface response can be developed around this feedback.

What is claimed is:

1. A method for directing the automatic transmission of images from an electronic camera, said method including the steps of:

storing network configuration information in a memory of the electronic camera;

using the network configuration information to connect the camera to an online service or internet service provider (ISP);

transferring pictures and data from the camera to the online service or ISP;

receiving information from the online service or ISP indicating that the transfer of the pictures and data is complete; and

displaying information on a display of the electronic camera indicating that the transfer of the pictures and data is complete;

wherein the network configuration information is generated at least in part in a host device adaptable for communication with the electronic camera, based on information previously stored in the host device and utilizable to connect the host device to the online service or ISP.

2. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes a PSTN (Public Switched Telephone Network).

3. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes an ISDN network.

4. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes an RF cellular phone network.

6

5. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes an Ethernet connection.

6. The method of claim 1 wherein the information previously stored in the host device comprises information stored in a network configuration preference file of the host device.

7. A method for directing the automatic transmission of images from an electronic camera, said method including the steps of:

storing network configuration information in a memory of the electronic camera, the network configuration information comprising information for using a communications network to establish a connection between the electronic camera and an online service or internet service provider (ISP); and

in response to user activation in the electronic camera of a transmission command specifying one or more pictures for transmission, automatically performing without further user intervention the steps of:

retrieving the stored network configuration information;

utilizing the retrieved information to establish a connection between the electronic camera and the online service or ISP;

transferring the one or more pictures from the electronic camera to the online service or ISP;

receiving information from the online service or ISP indicating that the transfer of the one or more pictures is complete; and

displaying the information on a display of the electronic camera indicating that the transfer of the one or more pictures is complete;

wherein the network configuration information is generated at least in part in a host device based on information previously stored in the host device, and the generated network configuration information is subsequently downloaded from the host device into the electronic camera in conjunction with the storing step.

8. The method of claim 7 wherein the host device comprises a personal computer.

9. The method of claim 7 wherein the information previously stored in the host device comprises information stored in a network configuration preference file of the host device.

* * * * *



US00662218B2

(12) **United States Patent**
Mighdoll et al.

(10) Patent No.: **US 6,662,218 B2**
(45) Date of Patent: ***Dec. 9, 2003**

(54) **METHOD OF TRANSCODING DOCUMENTS
IN A NETWORK ENVIRONMENT USING A
PROXY SERVER**

(75) Inventors: **Lee S. Mighdoll**, San Francisco, CA
(US); **Bruce A. Leak**, Palo Alto, CA
(US); **Stephen C. Perlman**, Mountain
View, CA (US); **Phillip Y. Goldman**,
Los Altos, CA (US)

(73) Assignee: **WebTV Networks, Inc.**, Mountain
View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 0 days.

This patent is subject to a terminal dis-
claimer.

(21) Appl. No.: **10/247,885**

(22) Filed: **Sep. 20, 2002**

(65) **Priority Publication Data**

US 2003/0014499 A1 Jan. 16, 2003

Related U.S. Application Data

(63) Continuation of application No. 09/343,067, filed on Jun.
29, 1999, now abandoned, which is a continuation of application
No. 08/656,924, filed on Jun. 3, 1996, now Pat. No.
5,918,013.

(51) **Int. Cl.** **G06F 15/16**

(52) **U.S. Cl.** **709/219; 709/203; 709/217;
709/228; 709/229; 709/246**

(58) **Field of Search** **709/200-203,
709/217-219, 227-229, 231-232, 246-247**

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,575,579 A 3/1986 Simon et al. 178/4
4,852,151 A 7/1989 Dittakavi et al. 379/97

4,922,523 A 5/1990 Hashimoto 379/96
4,975,944 A 12/1990 Cho 379/209
4,995,074 A 2/1991 Goldnan et al. 379/97
5,005,011 A 4/1991 Perlman et al. 340/728
5,095,494 A 3/1992 Takahashi et al. 375/10
5,220,420 A 6/1993 Hoarty et al. 358/86
5,241,587 A 8/1993 Horton et al. 379/92
5,263,084 A 11/1993 Chaput et al. 379/215
5,287,401 A 2/1994 Lin 379/98
5,299,307 A 3/1994 Young 395/161

(List continued on next page.)

OTHER PUBLICATIONS

Rosoff, Matt, Review: "Gateway Destination PC," c/net
inc., 2 pages, Feb. 19, 1996.

Seidman, Robert, Article: "What Larry and Lou Know (That
You Don't)," c/net inc., 2 pages, Jan. 29, 1996.

Stellin, Susan, Article: "The \$500 Web Box: Less is More?"
c/net inc., 2 pages, 1996.

(List continued on next page.)

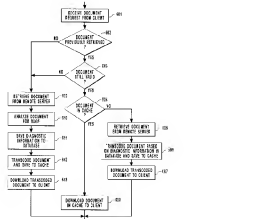
Primary Examiner—Bharat Barot

(74) Attorney, Agent, or Firm—Workman, Nydegger

(57) **ABSTRACT**

A method is described of providing a document to a client
coupled to a server. The server functions as a proxy on
behalf of the client for purposes of accessing a remote
server. In the method, a document is retrieved from the
remote server in response to a request from the client. The
document includes data to be used by the client in generating
a display. The proxying server alters (i.e., transcodes) the
data in the document to form a transcoded document. The
transcoded document is then transmitted to the client. The
proxying server transcodes the data in the document in order
to perform at least one of the following functions: (1)
matching decompression requirements at the client; (2)
converting the document into a format compatible for the
client; (3) reducing latency experienced by the client; and
(4) altering the document to fit into smaller memory space.

31 Claims, 12 Drawing Sheets



U.S. PATENT DOCUMENTS

5,325,423 A	6/1994	Lewis	379/60	5,678,041 A	10/1997	Baker et al.	395/609
5,329,619 A	7/1994	Page et al.	395/200	5,727,159 A	3/1998	Kikinis	395/200.76
5,341,293 A	8/1994	Verletney et al.	364/419.17	5,842,216 A	11/1998	Anderson et al.	707/203
5,369,688 A	11/1994	Tsukamoto et al.	379/100	5,889,955 A	3/1999	Shinozaki et al.	395/200.54
5,469,540 A	11/1995	Powers, III et al.	395/158	5,918,013 A	* 6/1999	Mighdoli et al.	709/217
5,488,411 A	1/1996	Lewis	348/8	5,978,817 A	11/1999	Gianandrea et al.	707/501
5,490,208 A	2/1996	Remillard	379/96	5,996,022 A	* 11/1999	Krueger et al.	709/247
5,530,852 A	6/1996	Meske, Jr. et al.	395/600	6,018,619 A	1/2000	Allard et al.	395/200.54
5,538,255 A	7/1996	Barker	463/41	6,226,412 B1	5/2001	Schwab	382/232
5,558,339 A	9/1996	Periman	463/42	OTHER PUBLICATIONS			
5,561,709 A	10/1996	Remillard	379/96	<i>Administrator's Guide, Netscape Proxy Server Version 2.0,</i>			
5,564,001 A	10/1996	Lewis	395/154	Netscape Communications Corporation, pp. 19-20, 1996.			
5,572,643 A	11/1996	Judson	395/793	Chankhuthod, Anawat, et al., "A Hierarchical Internet			
5,586,257 A	12/1996	Periman	463/42	<i>Object Cache,</i> " 1996 USEWIX Technical Conference (6			
5,586,260 A	12/1996	Hu	395/200.2	pages).			
5,612,730 A	3/1997	Lewis	348/8	Farrow, Rik, "Securing the Web: fire walls, proxy servers,			
5,623,600 A	4/1997	Ji et al.	395/187.01	<i>and data driven attacks,</i> " InfoWorld, Jun. 19, 1995, vol. 7,			
5,654,886 A	8/1997	Zereski, Jr. et al.	364/420	No. 25, pp. 103-104.			
5,657,390 A	8/1997	Elgarni et al.	380/49	* cited by examiner			
5,657,450 A	8/1997	Rao et al.	395/610				

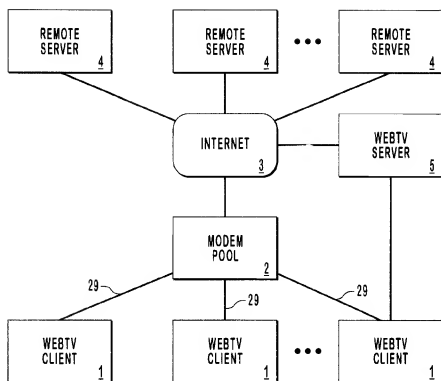


FIG. 1

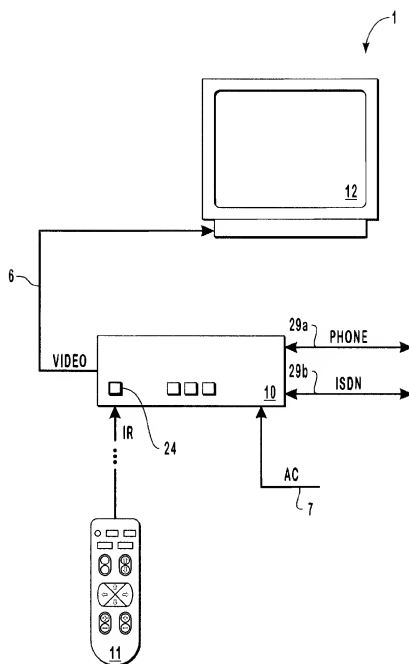


FIG. 2

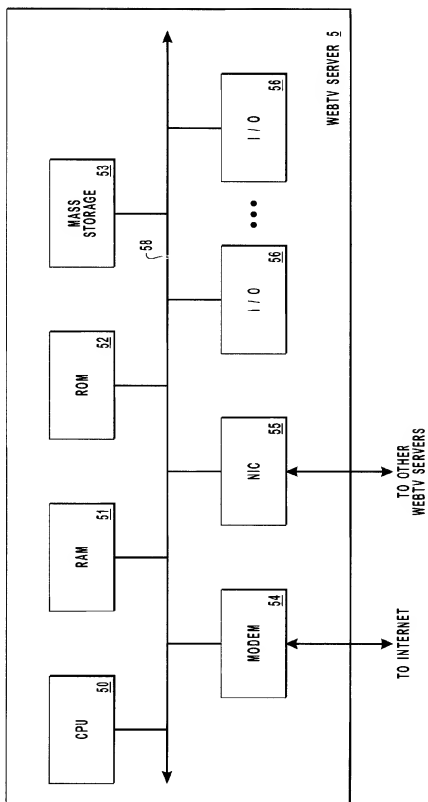


FIG. 3

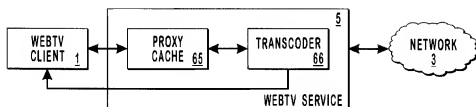


FIG. 4A

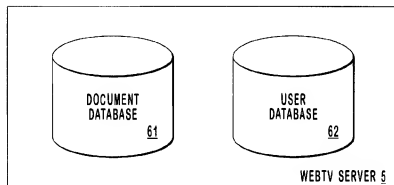


FIG. 4B

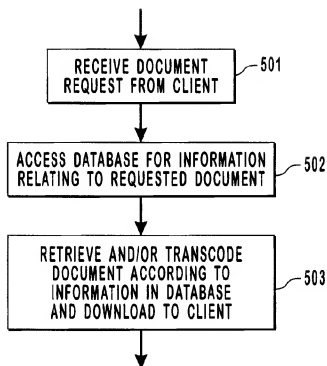


FIG. 5

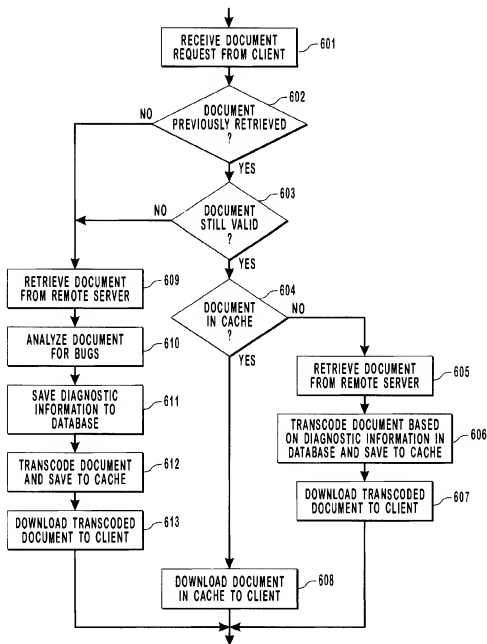


FIG. 6

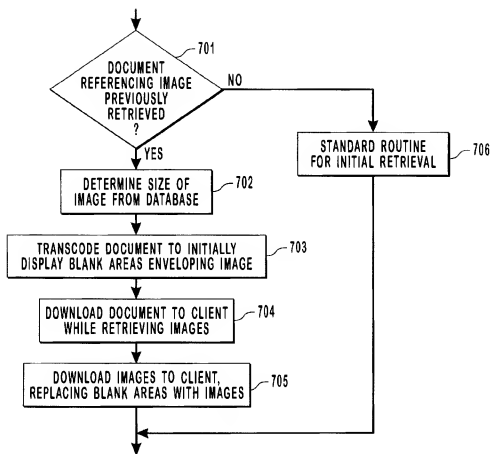


FIG. 7

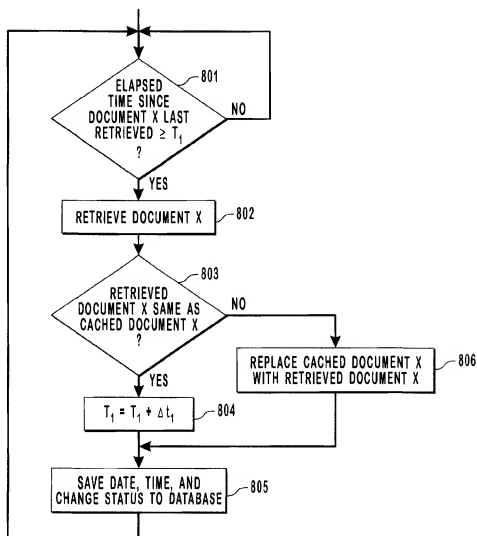


FIG. 8

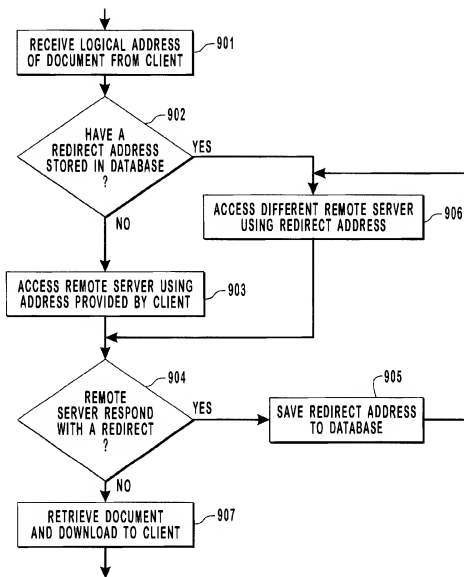


FIG. 9

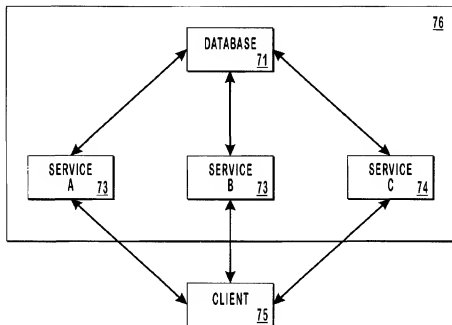


FIG. 10
(PRIOR ART)

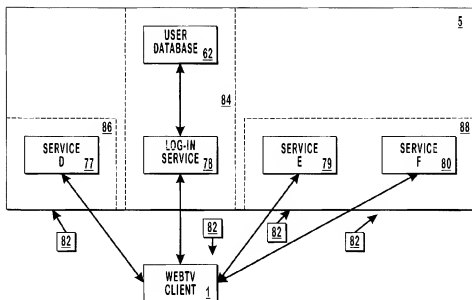


FIG. 11

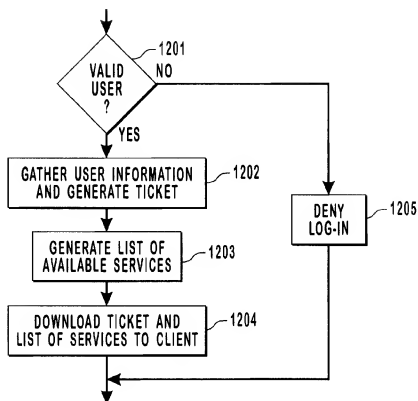


FIG. 12

1

METHOD OF TRANSCODING DOCUMENTS IN A NETWORK ENVIRONMENT USING A PROXY SERVER

RELATED APPLICATION

This is a continuation of U.S. patent application Ser. No. 09/343,067, entitled "Method of Transcoding Documents in a Network Environment Using a Proxy Server," filed Jun. 29, 1999, now abandoned which is a continuation of U.S. application Ser. No. 08/656,924 entitled "Method of Transcoding Documents in a Network Environment Using a Proxy Server," filed Jun. 3, 1996, now U.S. Pat. No. 5,918, 013 both of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. The Field of the Invention

The present invention pertains to the field of client-server computer networking. More particularly, the present invention relates to a method of transcoding documents in a network environment using a proxy server.

2. The Prior State of the Art

The number of people using personal computers has increased substantially in recent years, and along with this increase has come an explosion in the use of the Internet. One particular aspect of the Internet which has gained widespread use is the World-Wide Web ("the Web"). The Web is a collection of formatted hypertext pages located on numerous computers around the world that are logically connected by the Internet. Advances in network technology and software providing user interfaces to the Web ("Web browsers") have made the Web accessible to a large segment of the population. However, despite the growth in the development and use of the Web, many people are still unable to take advantage of this important resource.

Access to the Web has been limited thus far mostly to people who have access to a personal computer. However, many people cannot afford the cost of even a relatively inexpensive personal computer, while others are either unable or unwilling to learn the basic computer skills that are required to access the Web. Furthermore, Web browsers in the prior art generally do not provide the degree of user-friendliness desired by some people, and many computer novices do not have the patience to learn how to use the software. Therefore, it would be desirable to provide an inexpensive means by which a person can access the Web without the use of a personal computer. In particular, it would be desirable for a person to be able to access the Web pages using an ordinary television set and a remote control, so that the person feels more as if he or she is simply changing television channels, rather than utilizing a complex computer network.

Prior art Web technology also has other significant limitations which can make a person's experience unpleasant when browsing the Web. Web documents are commonly written in HTML (Hypertext Mark-up Language). HTML documents sometimes contain bugs (errors) or have features that are not recognized by certain Web browsers. These bugs or quirks in a document can cause a Web browser to fail. Thus, what is needed is a means for reducing the frequency with which client systems fail due to bugs or quirks in HTML documents.

Another problem associated with browsing the Web is latency. People commonly experience long, frustrating delays when browsing the Web. It is not unusual for a person to have to wait minutes after selecting a hypertext link for a

2

Web page to be completely downloaded to his computer and displayed on his computer screen. There are many possible causes for latency, such as heavy communications traffic on the Internet and slow response of remote servers. Latency can also be caused by Web pages including images. One reason for this effect is that, when an HTML document references an image, it takes time to retrieve the image itself after the referencing document has been retrieved. Another reason is that, in the prior art, if the referencing document does not specify the size of the image, the client system generally cannot display the Web page until the image itself has been retrieved. Numerous other sources of latency exist with respect to the Web. Therefore, what is needed is a means for reducing such latency, to eliminate some of the frustration which typically has been associated with browsing the Web.

Security is another concern associated with the Internet. Internet service providers (ISPs) generally maintain certain information about each customer in a database. This information may include information which a customer may not wish to become publicly known, such as social security numbers and credit card numbers. Maintaining the confidentiality of this information in a system that is connected to an expensive publicly-accessible computer network like the Internet can be problematic. Further, the problem can be aggravated by the fact that an ISP often provides numerous different services, each of which has access to this database. Allowing access to the database by many different entities creates many opportunities for security breaches to occur. Therefore, what is needed is a way to improve the security of confidential customer information in a server system coupled to the Internet.

SUMMARY AND OBJECTS OF THE INVENTION

A method is described of providing a document to a client coupled to a server. The server functions as a proxy on behalf of the client for purposes of accessing a remote server. In the method, a document is retrieved from the remote server in response to a request from the client. The document includes data to be used by the client in generating a display. The proxying server alters (i.e., transcodes) the data in the document to form a transcoded document. The transcoded document is then transmitted to the client.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follows. The proxying server transcodes the data in the document in order to perform at least one of the following functions: (1) matching decompression requirements at the client; (2) converting the document into a format compatible for the client; (3) reducing latency experienced by the client; and (4) altering the document to fit into smaller memory space.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follow.

BRIEF DESCRIPTION OF THE DRAWINGS

In order that the manner in which the above-recited and other advantages and objects of the invention are obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered limiting of its scope, the invention will be

3

described and explained with additional specificity and detail through the use of the accompanying drawings in which:

FIG. 1 illustrates several clients connected to a proxying server in a network;

FIG. 2 illustrates a client according to the present invention;

FIG. 3 is a block diagram of a server according to the present invention;

FIG. 4A illustrates a server including a proxy cache and a transcoder;

FIG. 4B illustrates databases used in a server according to the present invention;

FIG. 5 is a flow diagram illustrating a routine for transcoding a document retrieved from a remote server using data stored in a persistent database;

FIG. 6 is a flow diagram illustrating a routine for transcoding an HTML document for purposes of eliminating bugs or undesirable features;

FIG. 7 is a flow diagram illustrating a routine for reducing latency when downloading a document referencing an image to a client;

FIG. 8 is a flow diagram illustrating a routine for updating documents stored in the proxy cache using data stored in a persistent database;

FIG. 9 is a flow diagram illustrating a routine used by a server for retrieving documents from another remote server;

FIG. 10 is a block diagram of a prior art server system showing a relationship between various services and a database;

FIG. 11 is a block diagram of a server system according to the present invention showing a relationship between various services and a user database; and

FIG. 12 is a flow diagram illustrating a routine used by a server for regulating access to various services provided by the server.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

A method and apparatus are described for providing proxying and transcoding of documents in a network. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be evident, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

The present invention includes various steps, which will be described below. The steps can be embodied in machine-executable instructions, which can be used to cause a general-purpose or special-purpose processor programmed with the instructions to perform the steps. Alternatively, the steps of the present invention might be performed by specific hardware components that contain hardwired logic for performing the steps, or by any combination of programmed computer components and custom hardware components.

I. System Overview

The present invention is included in a system, known as WebTV™, for providing a user with access to the Internet. A user of a WebTV™ client generally accesses a WebTV™ server via a direct-dial telephone (POTS, for "plain old

4

telephone service"), ISDN (Integrated Services Digital Network), or other similar connection, in order to browse the Web, send and receive electronic mail (e-mail), and use various other WebTV™ network services. The WebTV™ network services are provided by WebTV™ servers using software residing within the WebTV™ servers in conjunction with software residing within a WebTV™ client.

FIG. 1 illustrates a basic configuration of the WebTV™ network according to one embodiment. A number of WebTV™ clients 1 are coupled to a modem pool 2 via direct-dial, bi-directional data connections 29, which may be telephone (POTS, i.e., "plain old telephone service"), ISDN (Integrated Services Digital Network), or any other similar type of connection. The modem pool 2 is coupled typically through a router, such as that conventionally known in the art, to a number of remote servers 4 via a conventional network infrastructure 3, such as the Internet. The WebTV™ system also includes a WebTV™ server 5, which specifically supports the WebTV™ clients 1. The WebTV™ clients 1 each have a connection to the WebTV™ server 5 either directly or through the modem pool 2 and the Internet 3. Note that the modem pool 2 is a conventional modem pool, such as those found today throughout the world providing access to the Internet and private networks.

Note that in this description, in order to facilitate explanation the WebTV™ server 5 is generally discussed as if it were a single device, and functions provided by the WebTV™ services are generally discussed as being performed by such single device. However, the WebTV™ server 5 may actually comprise multiple physical and logical devices connected in a distributed architecture, and the various functions discussed below which are provided by the WebTV™ services may actually be distributed among multiple WebTV™ server devices.

II. Client System

FIG. 2 illustrates a WebTV™ client 1. The WebTV™ client 1 includes an electronics unit 10 (hereinafter referred to as "the WebTV™ box 10"), an ordinary television set 12, and a remote control 11. In an alternative embodiment of the present invention, the WebTV™ box 10 is built into the television set 12 as an integral unit. The WebTV™ box 10 includes hardware and software for providing the user with a graphical user interface, by which the user can access the WebTV™ network services, browse the Web, send e-mail, and otherwise access the Internet.

The WebTV™ client 1 uses the television set 12 as a display device. The WebTV™ box 10 is coupled to the television set 12 by a video link 6. The video link 6 is an RF (radio frequency), S-video, composite video, or other equivalent form of video link. In the preferred embodiment, the client 1 includes both a standard modem and an ISDN modem, such that the communication link 29 between the WebTV™ box 10 and the server 5 can be either a telephone (POTS) connection 29a or an ISDN connection 29b. The WebTV™ box 10 receives power through a power line 7.

Remote control 11 is operated by the user in order to control the WebTV™ client 1 in browsing the Web, sending e-mail, and performing other Internet-related functions. The WebTV™ box 10 receives commands from remote control 11 via an infrared (IR) communication link. In alternative embodiments, the link between the remote control 11 and the WebTV™ box 10 may be RF or any equivalent mode of transmission.

III. Server System

The WebTV™ server 5 generally includes one or more computer systems generally having the architecture illus-

5

trated in FIG. 3. It should be noted that the illustrated architecture is only exemplary, the present invention is not constrained to this particular architecture. The illustrated architecture includes a central processing unit (CPU) 50, random access memory (RAM) 51, read-only memory (ROM) 52, a mass storage device 53, a modem 54, a network interface card (NIC) 55, and various other input/output (I/O) devices 56. Mass storage device 53 includes a magnetic, optical, or other equivalent storage medium. I/O devices 56 may include any or all of devices such as a display monitor, keyboard, cursor control device, etc. Modem 54 is used to communicate data to and from remote servers 4 via the Internet.

As noted above, the WebTV™ server 5 may actually comprise multiple physical and logical devices connected in a distributed architecture. Accordingly, NIC 55 is used to provide data communication with other devices that are part of the WebTV™ services. Modem 54 may also be used to communicate with other devices that are part of the WebTV™ services and which are not located in close geographic proximity to the illustrated device.

According to the present invention, the WebTV™ server 5 acts as a proxy in providing the WebTV™ client 1 with access to the Web and other WebTV™ services. More specifically, WebTV™ server 5 functions as a "caching proxy." FIG. 4A illustrates the caching feature of the WebTV™ server 5. In FIG. 4A, the WebTV™ server 5 is functionally located between the WebTV™ client 1 and the Internet infrastructure 3. The WebTV™ server 5 includes a proxy cache 65 which is functionally coupled to the WebTV™ client 1. The proxy cache 65 is used for temporary storage of Web documents, images, and other information which is frequently used by either the WebTV™ client 1 or the WebTV™ server 5.

A document transcoder 66 is functionally coupled between the proxy cache 65 and the Internet infrastructure 3. The document transcoder 66 includes software which is used to automatically revise the code of Web documents retrieved from the remote servers 4, for purposes which are described below.

The WebTV™ service provides a document database 61 and a user database 62, as illustrated in FIG. 4B. The user database 62 contains information that is used to control certain features relating to access privileges and capabilities of the user of the client 1. This information is used to regulate initial access to the WebTV™ service, as well as to regulate access to the individual services provided by the WebTV™ system, as will be described below. The document database 61 is a persistent database which stores certain diagnostic and historical information about each document and image retrieved by the server 5, as is now described.

A. Document Database

The basic purpose of the document database 61 is that, after a document has once been retrieved by the server 5, the stored information can be used by the server 5 to speed up processing and downloading of that document in response to all future requests for that document. In addition, the transcoding functions and various other functions of the WebTV™ service are facilitated by making use of the information stored in the document database 61, as will be described below.

Referring now to FIG. 5, the server 5 initially receives a document request from a client 1 (step 501). The document request will generally result from the user of the client 1

6

activating a hypertext anchor (link) on a Web page. The act of activating a hypertext anchor may consist of clicking on underlined text in a displayed Web page using a mouse, for example. The document request will typically (but not always) include the URL (Uniform Resource Locator) or other address of the selected anchor. Upon receiving the document request, the server 5 optionally accesses the document database 61 to retrieve stored information relating to the requested document (step 502). It should be noted that the document database 61 is not necessarily accessed in every case. The information retrieved from the document database 61 is used by the server 5 for determining, among other things, how long a requested document has been cached and/or whether the document is still valid. The criteria for determining validity of the stored document are discussed below. The server 5 retrieves the document from the cache 65 if the stored document is valid; otherwise, the server 5 retrieves the document from the appropriate remote server 4 (step 503). The server 5 automatically transcodes the document as necessary based on the information stored in the document database 61 (step 503). The transcoding functions are discussed further below.

The document database 61 includes certain historical and diagnostic information for every Web page that is accessed at any time by a WebTV™ client 1. As is well known, a Web page may correspond to a document written in a language such as HTML (Hypertext Mark-Up Language), VRML (Virtual Reality Modelling Language), or another suitable language. Alternatively, a Web page may represent an image, or a document which references one or more images. According to the present invention, once a document or image is retrieved by the WebTV™ server 5 from a remote server 4 for the first time, detailed information on this document or image is stored permanently in the document database 61. More specifically, for every Web page that is retrieved from a remote server 4, any or all of the following data are stored in the document database 61:

- 1) information identifying bugs (errors) or quirks in the Web page, or undesirable effects caused when the Web page is displayed by a client 1;
- 2) relevant bug-finding algorithms;
- 3) the date and time the Web page was last retrieved;
- 4) the date and time the Web page was most recently altered by the author;
- 5) a checksum for determining whether the Web page has been altered;
- 6) the size of the Web page (in terms of memory);
- 7) the type of Web page (e.g., HTML document, image, etc.);
- 8) a list of hypertext anchors (links) in the Web page and corresponding URLs;
- 9) a list of the most popular anchors based on the number of "hits" (requests from a client 1);
- 10) a list of related Web pages which can be prefetched;
- 11) whether the Web page has been redirected to another remote server 4;
- 12) a redirect address (if appropriate);
- 13) whether the redirect (if any) is temporary or permanent, and if permanent, the duration of the redirect;
- 14) if the Web page is an image, the size of the image in terms of both physical dimensions and memory space;
- 15) the sizes of in-line images (images displayed in text) referenced by the document defining the Web page;

7

- 16) the size of the largest image referenced by the document;
- 17) information identifying any image maps in the Web page;
- 18) whether to resize any images corresponding to the Web page;
- 19) an indication of any forms or tables in the Web page;
- 20) any unknown protocols;
- 21) any links to "dead" Web pages (i.e., pages which are no longer active);
- 22) the latency and throughput of the remote server 4 on which the Web page is located;
- 23) the character set of the document;
- 24) the vendor of the remote server 4 on which the Web page is located;
- 25) the geographic location of the remote server 4 on which the Web page is located;
- 26) the number of other Web pages which reference the subject Web page;
- 27) the compression algorithm used by the image or document;
- 28) the compression algorithm chosen by the transcoder;
- 29) a value indicating the popularity of the Web page based on the number of hits by clients; and
- 30) a value indicating the popularity of other Web pages which reference the subject Web page.

B. Transcoding

As mentioned above, the WebTV™ services provide a transcoder 66, which is used to rewrite certain portions of the code in an HTML document for various purposes. These purposes include: (1) correcting bugs in documents; (2) correcting undesirable effects which occur when a document is displayed by the client 1; (3) improving the efficiency of transmission of documents from the server 5 to the client 1; (4) matching hardware decompression technology within the client 1; (5) resizing images to fit on the television set 12; (6) converting documents into other formats to provide compatibility; (7) reducing latency experienced by a client 1 when displaying a Web page with in-line images (images displayed in text); and, (8) altering documents to fit into smaller memory spaces.

There are three transcoding modes used by the transcoder 66: (1) streaming, (2) buffered, and (3) deferred. Streaming transcoding refers to the transcoding of documents on a line-by-line basis as they are retrieved from a remote server 4 and downloaded to the client 1 (i.e., transcoding "on the fly"). Some documents, however, must first be buffered in the WebTV™ server 5 before transcoding and downloading them to the client 1. A document may need to be buffered before transmitting it to the client 1 if the type of changes to be made can only be made after the entire document has been retrieved from the remote server 4. Because the process of retrieving and downloading a document to the client 1 increases latency and decreases throughput, it is not desirable to buffer all documents. Therefore, the transcoder 66 accesses and uses information in the document database 61 relating to the requested document to first determine whether a requested document must be buffered for purposes of transcoding, before the document is retrieved from the remote server 4.

In the deferred mode, transcoding is deferred until after a requested document has been downloaded to a client 1. The deferred mode therefore reduces latency experienced by the

8

client 1 in receiving the document. Transcoding may be performed immediately after downloading or any time thereafter. For example, it may be convenient to perform transcoding during periods of low usage of WebTV™ services, such as at night. This mode is useful for certain types of transcoding which are not mandatory.

1. Transcoding for Bugs and Quirks

One characteristic of some prior art Web browsers is that they may experience failures ("crashes") because of bugs or unexpected features ("quirks") that are present in a Web document. Alternatively, quirks in a document may cause an undesirable result, even though the client does not crash. Therefore, the transcoding feature of the present invention provides a means for correcting certain bugs and quirks in a Web document. To be corrected by the transcoder 66, bugs and quirks must be identifiable by software running on the server 5. Consequently, the transcoder 66 will generally only correct conditions which have been previously discovered, such as those discovered during testing or reported by users. Once a bug or quirk is discovered, however, algorithms are added to the transcoder 66 to both detect the bug or quirk in the future in any Web document and to automatically correct it.

There are countless possibilities of bugs or quirks which might be encountered in a Web document. Therefore, no attempt will be made herein to provide an exhaustive list. Nonetheless, some examples may be useful at this point. Consider, for example, an HTML document that is downloaded from a remote server 4 and which contains a table having a width specified in the document as "0." This condition might cause a failure if the client were to attempt to display the document as written. This situation therefore, can be detected and corrected by the transcoder 66. Another example is a quirk in the document which causes quotations to be terminated with too many quotation marks. Once the quirk is first detected and an algorithm is written to recognize it, the transcoder 66 can automatically correct the quirk in any document.

If a given Web document has previously been retrieved by the server 5, there will be information regarding that document available in the document database 61 as described above. The information regarding this document will include whether or not the document included any bugs or quirks that required transcoding when the document was previously retrieved. The transcoder 66 utilizes this information to determine whether (1) the document is free of bugs and quirks, (2) the document has bugs or quirks which can be remedied by transcoding on the fly, or (3) the document has bugs or quirks which cannot be corrected on the fly (i.e., buffering is required).

FIG. 6 illustrates a routine for transcoding a Web document for purposes of eliminating bugs and quirks. Initially, the server 5 receives a document request from the client 1 (step 601). Next, the document database 61 is accessed to determine whether or not the requested document has been previously retrieved (step 602). If the document has not been previously retrieved, then the server 5 retrieves the document from the remote server 4 (step 609). Next, the retrieved document is analyzed for the presence of bugs or unusual conditions (step 610). Various diagnostic information is then stored in the document database 61 as a result of the analysis to note any bugs or quirks that were found (step 611). If any bugs or quirks were found which can be corrected by the transcoder 66, the document is then transcoded and saved to the proxy cache 65 (step 612). The transcoded document is

9

then downloaded to the client 1 (step 613). It should be noted that transcoding can be deferred until after the document has been downloaded, as described above; hence, the sequence of FIG. 6 is illustrative only.

If (in step 602) the requested document had been previously retrieved, then it is determined whether the requested document is still valid (step 603) and whether the document is present in the proxy cache 65 (step 604). If the document is no longer valid, then the document is retrieved from the remote server 4, analyzed for bugs and quirks, transcoded as required, and then downloaded to the client 1 as described above (steps 609-613). Methods for determining validity of a document are discussed below. If the document is still valid (step 603) and the document is present in the cache 65, the document is downloaded to the client 1 in its current form (as it is stored in the cache), since it has already been transcoded (step 608).

The document, however, may be valid but not present in the cache. This may be the case, for example, if the document has not been requested recently and the cache 65 has become too full to retain the requested document. In that case, the document is retrieved again from the remote server 4 (step 605) and then transcoded on the basis of the previously-acquired diagnostic information stored within the database 61 for that document. The document is then saved to the cache 65 (step 606). Note that because the document is still valid, it is assumed that the diagnostic information stored in the document database 61 for that document is still valid and that the transcoding can be performed on the basis of that information. Accordingly, once the document is transcoded, the transcoded document is downloaded to the client 1 (step 607). Again, note that transcoding can be deferred until after the document has been downloaded in some cases.

The validity of the requested document can be determined based on various different criteria. For example, some HTML documents specify a date on which the document was created, a length of time for which the document will be valid, or both. The validity determination can be based upon such information. For example, a document which specifies only the date of creation can be automatically deemed invalid after a predetermined period of time has passed.

Alternatively, validity can be based upon the popularity of the requested document. "Popularity" can be quantified based upon the number of hits for that document, which is tracked in the document database 61. For example, it might be prudent to simply assign a relatively short period of validity to a document which is very popular and a longer period of validity to a document which is less popular.

Another alternative basis for the validity of a document is the observed rate of change of the document. Again, data in the persistent document database 61 can be used. That is, because the document database 61 stores the date and time on which the document was last observed to change, the server 5 can approximate how often the document actually changes. A document or image which is observed to change frequently (e.g., a weather map or a news page) can be assigned a relatively short period of validity. It will be recognized that numerous other ways of determining validity are possible.

2. Transcoding to Reduce Latency

Another purpose for transcoding is to allow documents requested by a client 1 to be displayed by the client 1 more rapidly. Many HTML documents contain references to "in-line" images, or images that will be displayed in text in a

10

Web page. The normal process used in the prior art to display a Web page having in-line images is that the HTML document referencing the image is first downloaded to the client, followed by the client's requesting the referenced image. The referenced image is then retrieved from the remote server on which it is located and downloaded to the client. One problem associated with the prior art, however, is that the speed with which a complete Web page can be displayed to the user is often limited by the time it takes to retrieve in-line images. One reason for this is that it simply takes time to retrieve the image itself after the referencing document has been retrieved. Another reason is that, in the prior art, if the referencing document does not specify the size of the image, the Web page generally cannot be displayed until the image itself has been retrieved. The present invention overcomes these limitations.

According to the present invention, information stored in the document database 61 regarding the in-line images is used to transcode the referencing document in order to reduce latency in displaying the Web page. Once any document which references an in-line image is initially retrieved by the server 5, the fact that the document references an in-line image is stored in the document database 61. In addition, the size of the image is determined, either from the document (if specified) or from the image itself, and then stored in the document database 61. Consequently, for documents which do not specify the size of their in-line images, the size information stored in the database 61 is then used the next time the document is requested in order to reduce latency in downloading and displaying the Web page.

Refer now to FIG. 7, which illustrates a routine for reducing latency when downloading a document referencing an image to a client 1. Assume that a client 1 sends a request to the server 5 for an HTML document containing a reference to an in-line image. Assume further that the size of the image is not specified in the document itself. Initially, the server 5 determines whether that document has been previously retrieved (step 701). If not, the standard initial retrieval and transcoding procedure is followed (step 706), as described in connection with FIG. 6. If, however, the document has been previously retrieved, then the transcoder 66 accesses the size information stored in the document database 61 for the in-line image (step 702). Based on this size information, the HTML document is transcoded such that, when the Web page is initially displayed by the client 1, the area in which the image belongs is replaced by a blank region enveloping the shape of the image (step 703). Thus, any in-line image referenced by a document is displayed initially as a blank region. Consequently, the client 1 can immediately display the Web page corresponding to the HTML document even before the referenced image has been retrieved or downloaded (i.e., even before the size of the image is known to the client 1).

As the transcoded HTML document is downloaded to the client, the image is retrieved from the appropriate remote server 4 (step 704). Once the image is retrieved from the remote server 4 and downloaded to the client 1, the client 1 replaces the blank area in the Web page with the actual image (step 705).

3. Transcoding to Display Web Pages on a Television

As noted above, the client 1 utilizes an ordinary television set 12 as a display device. However, images in Web pages are generally formatted for display on a computer monitor, not a television set. Consequently, the transcoding function

11

of the present invention is used to resize images for display on the television set 12. This includes resealing images as necessary to avoid truncation when displayed on the television set 12.

It should be noted that prior art Web browsers which operate on computer monitors typically use resizable windows. Hence, the size of the visible region varies from client to client. However, because the web browser used by the WebTV™ client 1 is specifically designed for display on a television set, the present invention allows documents and images to be formatted when they are cached.

4. Transcoding for Transmission Efficiency

Documents retrieved by the server 5 are also transcoded to improve transmission efficiency. In particular, documents can be transcoded in order to reduce high frequency components in order to reduce interface flicker when they are displayed on a television set. Various methods for coding software or hardware to reduce perceptual interface flicker are described in co-pending U.S. patent application Ser. No. 08/656,923, filed on Jun. 3, 1996.

Documents can also be transcoded in order to lower the resolution of the displayed Web page. Reducing the resolution is desirable, because images formatted for computer systems will generally have a higher resolution than the NTSC (National Television Standards Committee) video format used by conventional television sets. Since the NTSC video does not have the bandwidth to reproduce the resolution of computer-formatted images, the bandwidth consumed in transmitting images to the client 1 at such a high resolution would be wasted.

5. Other Uses for Transcoding

Transcoding is also used by the present invention to recode a document using new formats into older, compatible formats. Images are often displayed in the JPEG (Joint Picture Experts Group) format or the GIF image format. JPEG often consumes less bandwidth than GIF, however. Consequently, images which are retrieved in GIF format are sometimes transcoded into JPEG format. Methods for generally converting images between GIF and JPEG formats are well known.

Other uses for transcoding include transcoding audio files. For example, audio may be transcoded into different formats in order to achieve a desired balance between memory usage, sound quality, and data transfer rate. In addition, audio may be transcoded from a file format (e.g., an "AU" file) to a streaming format (e.g., MPEG 1 audio). Yet another use of audio transcoding is the transcoding of MIDI (Musical Instrument Digital Interface) data to streaming variants of MIDI.

Additionally, documents or images requiring a large amount of memory (e.g., long lists) can be transcoded in order to consume less memory space in the client 1. This may involve, for example, separating a large document or image into multiple sections. For example, the server 5 can insert tags at appropriate locations in the original document so that the document appears to the client 1 as multiple Web pages. Hence, while viewing a given page representing a portion of the original document, the user can view the next page (i.e., the next portion of the original document) by activating a button on the screen as if it were an ordinary hypertext anchor.

C. Proxying

As noted above, the server 5 functions as a proxy on behalf of the client 1 for purposes of accessing the Web. The

12

document database 61 is used in various ways to facilitate this proxy role, as will now be described.

1. Updating Cached Documents

It is desirable to store frequently-requested HTML documents and images in the proxy cache 65 to further reduce latency in providing Web pages to the client 1. However, because some documents and images change over time, documents in the cache 65 will not be valid indefinitely, as mentioned above. A weather map or a news-related Web page, for example, are likely to be updated quite frequently. Consequently, it is desirable for the server 5 to have the ability to estimate the frequency with which documents change, in order to determine how long a document can safely remain within the proxy cache 65 without being updated.

The persistent database 65 is used to store the date and time of the last several fetches of each document and image retrieved from a remote server 4, along with an indication of any changes that were detected, if any. A document or image which has been stored in the cache 65 is then retrieved on a periodic basis to determine if it has been changed. Change status information indicating whether the document has changed since the previous fetch is then stored in the document database 61. If no changes are detected, then the time interval between fetches of this document is increased. If the document has changed, the time interval is maintained or decreased. As a result, items in the cache 65 which change frequently will be automatically updated at frequent intervals, whereas documents which do not change often will be replaced in the cache less frequently.

FIG. 8 illustrates a routine for updating documents stored in the proxy cache 65 using data stored in the document database 61. Assume a document X has been stored in the proxy cache 65. Document X remains in the cache 65 until a predetermined update period T_1 expires (step 801). Upon the expiration of the update period T_1 , the document X is again retrieved from the appropriate remote server 4 (step 802). The newly-retrieved document X is then compared to the cached version of document X (step 803). If the document has changed, then the cached version of document X is replaced with the newly-retrieved version of document X (step 806). If not, then the update period T_1 is increased according to a predetermined time increment ΔT_1 (step 804). In any case, the date and time and the change status of document X is saved to the document database 61 (step 805).

2. Document and Image Prefetching

The document database 61 is also used by the server 5 to store prefetching information relating to documents and images. In particular, the database stores, for each document that has been retrieved, a list of images referenced by the document, if any, and their locations. Consequently, the next time a document is requested by a client 1, the images can be immediately retrieved by the server 5 (from the cache 65, if available, or from the remote server 4), even before the client 1 requests them. This procedure improves the speed with which requested Web pages are downloaded to the client.

The document database 61 is also used to facilitate a process referred to as "server-advised client prefetching." Server-advised client prefetching allows the server 5 to inform the client 1 of documents or images which are popular to allow the client 1 to perform the prefetching. In particular, for any given document, a list is maintained in the

13

server 5 of the most popular hypertext anchors in that document (i.e., those which have previously received a large number of hits). When that document is requested by the client 1, the server 5 provides the client 1 with an indication of these popular links.

3. Redirects

Web pages are sometimes forwarded from the remote server on which they are initially placed to a different location. Under the HTTP (Hypertext Transport Protocol), such forwarding is sometimes referred to as a "redirect." When an HTML document is initially stored on one remote server and then later transferred to another remote server, the first remote server will provide, in response to a request for that document, an indication that the document has been transferred to a new remote server. This indication generally includes a forwarding address ("redirect address"), which is generally a URL.

In the prior art, when a computer requesting a Web page receives a redirect, it must then submit a new request to the redirect address. Having to submit a second request and wait for a second response consumes time and increases overall latency. Consequently, the present invention uses the document database 61 to store any redirect address for each document or image. Any time a redirected document is requested, the server 5 automatically accesses the redirect address to retrieve the document. The document or image is provided to the client 1 based on only a single request from the client 1. The change in location of the redirected document or image remains completely transparent to the client 1.

FIG. 9 illustrates a routine performed by the server 5 in accessing documents which may have been forwarded to a new remote server. Initially, the server 5 receives a request for a document, which generally includes an address (step 901). The server 5 then accesses the document database 61 to determine whether there is a redirect address for the requested document (step 902). If there is no redirect address, then the server 5 accesses a remote server 4 based on the address provided in the document request from the client 1 (step 903). Assuming that the remote server 4 does not respond to the server 5 with a redirect (step 904), the document is retrieved and downloaded to the client 1 by the server 5 (step 907). If, however, a redirect address was stored in the document database 61 (step 902), then the server 5 accesses the requested document according to the redirect address (step 906). Or, if the remote server 4 responded with a redirect (step 904), then the server 5 saves the redirect address to the document database 61 (step 905) and accesses the requested document according to the redirect address (step 906).

4. Other Proxy Functions

The document database 61 also stores information relating to the performance of each remote server 4 from which a document is retrieved. This information includes the latency and throughput of the remote server 4. Such information can be valuable in instances where a remote server 4 has a history of responding slowly. For example, when the document is requested, this knowledge can be used by the server 5 to provide a predefined signal to the client 1. The client 1 can, in response to the signal, indicate to the user that a delay is likely and give the user the option of canceling the request.

5. Backoff Mode

Although the server 5 generally operates in the proxy mode, it can also enter a "backoff mode" in which the server

14

5 does not act as a proxy, or the server 5 performs only certain aspects of the normal proxying functions. For example, if the proxy cache 65 is overloaded, then the server 5 can enter a backoff mode in which documents are not cached but are transcoded as required. Alternatively, during times when the server 5 is severely overloaded with network traffic, the server 5 may instruct the client 1 to bypass the server 5 and contact remote servers 4 directly for a specified time or until further notice. Or, the server 5 can enter a flexible backoff mode in which the client 1 will be instructed to contact a remote server 4 directly only for certain Web sites for a limited period of time.

D. Access to WebTV™ Services

The WebTV™ server 5 provides various services to the client 1, such as proxying and electronic mail ("e-mail"). In the prior art, certain difficulties are associated with allowing a client computer access to different services of an Internet service, as will now be explained with reference to FIG. 10.

FIG. 10 illustrates a client-server system according to one prior art embodiment. The server 76 provides various services A, B, and C. The server 76 includes a database 71 for storing information on the user's access privileges to services A, B, and C. The client 75 of the embodiment of FIG. 10 accesses any of services A, B, and C by contacting that service directly. The contacted service then accesses the database 71, which stores the access privileges of the client 75, to determine whether the client 75 should be allowed to access that service. Hence, each service provided by the server 76 requires direct access to the database 71. This architecture results in a large number of accesses being made to the database 71, which is undesirable. In addition, the fact that each service independently has access to the database 71 raises security concerns. Specifically, it can be difficult to isolate sensitive user information. The present invention overcomes such difficulties using a technique which is now described.

1. Tickets Containing Privileges And Capabilities

As shown in FIG. 11, the server 5 provides a number of services D, E, and F 77, 79, and 80, respectively, and a log-in service 78. The log-in service 78 is used specifically to control initial log-on procedures by a client 1. The log-in service 78 has exclusive access to the user database 62 (discussed above with respect to FIG. 4B). The log-in service 78 and the user database 62 are located within a first security zone 84. Service D is located within a second security zone 86, while services E and F are contained within a third security zone 88. Note that the specific arrangement of security zones 84, 86, and 88 with respect to services D, E, and F is illustrative only.

The user database 62 of the present invention stores various information pertaining to each authorized user of a client 1. This information includes account information, a list of the WebTV™ services that are available to the particular user, and certain user preferences. For example, a particular user may not wish his client 1 to be used to access Web pages having adult-oriented subject matter. Consequently, the user would request that his account be filtered to prevent access to such material. This request would then be stored as part of the user data in the user database 62.

With regard to user preferences, the hypertext links selected by a given user can be tracked, and those having the largest number can be stored in the user database 62. The list can then be provided to the client 1 for use in generating a

15

menu screen of the user's favorite Web sites, to allow the user to directly access those Web sites. The list can also be used by the server 5 to analyze the user's interests and to formulate and provide to the user a list of new Web sites which the user is likely to be interested in. The list might be composed by associated key words in Web pages selected by the user with other Web pages.

Referring again to FIG. 11, in response to a log-on request by a client 1, the log-in service 78 consults the user database 62 to determine if access to the server 5 by this particular client 1 is authorized. Assuming access is authorized, the log-in service 78 retrieves certain user information pertaining to this particular client 1 from the user database 62. The log-in service then generates a "ticket" 82, which is an information packet including the retrieved information. The ticket 82 is then provided to the client 1 which requested access.

The ticket 82 includes all information necessary to describe the access privileges of a particular user with respect to all services provided by the server 5. For example, the ticket may include the user name registered to the client 1, the e-mail address assigned to client 1, and any filtering requested by the user with respect to viewing Web sites. Each time the user requests access to one of the services D, E, or F, the client 1 submits a copy of the ticket 82 to that service. The requested service can then determine from the copy of the ticket 82 whether access to that service by that client 1 is authorized and, if so, any important information relating to such access.

None of the services provided by the server 5, other than the log-in service 78, has access to the user database 62. Hence, any security-sensitive information can be isolated within the user database 62 and the log-in service 78. Such isolation allows the individual services provided by the server 5 to be placed within separate "firewalls" (security regions), illustrated as security zones 84, 86, and 88. In addition, this technique greatly reduces the number of accesses required to the user database 62 compared to the prior art embodiment illustrated in FIG. 10.

2. Redundancy of Services and Load Balancing

The present invention also includes certain redundancies in the various services provided by the server 5. In particular, a given service (e.g., e-mail) can be provided by more than one physical or logical device. Each such device is considered a "provider" of that service. If a given provider is overloaded, or if the client 1 is unable to contact that provider, the client 1 can contact any of the other providers of that service. When the server 5 receives a log-in request from a client 1, in addition to generating the above-described ticket 82, the log-in service 78 dynamically generates a list of available WebTV™ services and provides this list to the client 1.

The server 5 can update the list of services used by any client 1 to reflect services becoming unavailable or services coming on-line. Also, the list of services provided to each client 1 can be updated by the server 5 based upon changes in the loading of the server 5, in order to optimize traffic on the server 5. In addition, a client's list of services can be updated by services other than the log-in service 78, such that one service can effectively introduce another service to the client 1. For example, the e-mail service may provide a client 1 with the name, port number and IP of its address book service. Thus, one service can effectively, and securely within the same chain of trust, introduce another service to the client 1.

16

This list of services includes the name of each service, a port number for the provider of each service, and an IP (Internet Protocol) for each service. Different providers of the same service are designated by the same name, but different port numbers and/or IPs. Note that in a standard URL, the protocol is normally specified at the beginning of the URL, such as "HTTP://www. . . ." under the HTTP protocol. However, according to the present invention, the normal protocol designation (i.e., "HTTP") in the URL is replaced with the name of the service, since the port number and IP for each service are known to the client 1. Hence, the client 1 can access any of the redundant providers of a given service using the same URL. This procedure effectively adds a level of indirection to all accesses made to any WebTV™ service and automatically adds redundancy to the proxy service. It should also be noted that separate service names can also refer to the same service.

Assume, for example, that the e-mail service provided by the WebTV™ system is designated by the service name "WTV-mailto." A client 1 can access any provider of this e-mail service using the same URL. The client 1 merely chooses the appropriate port number and IP number to distinguish between providers. If the client 1 is unable to connect to one e-mail provider, it can simply contact the next one in the list.

Thus, at log-in time, a client 1 is provided with both a ticket containing privileges and capabilities as well as a list of service providers, as illustrated in FIG. 12. Initially, the log-in service 78 determines whether the user of client 1 is a valid user (step 1201). If not, log-in is denied (step 1205). If the user is a valid user, then the log-in service 78 gathers user information from the user database 62 and generates a ticket 82 (step 1202). The log-in service 78 also generates the above-described list of services (step 1203). The ticket 82 and the list of services are then downloaded to the client 1 (step 1204).

3. Asynchronous Notification to Clients by Server

Another limitation associated with prior art Internet servers is the inability to provide asynchronous notification information to the client in the absence of a request from the client to do so. It would be desirable, for example, for a server to notify a client on its own initiative when a particular Web page has changed or that a particular service is inaccessible. The server 5 of the present invention provides such capability, and the client 1 is configured to receive and decode such notifications. For example, the client 1 can receive updates of its listing of service providers from the server 5 at various points in time, as already described. Similarly, if a particular service provider becomes unavailable, that fact will be automatically communicated to the client 1. As another example, if e-mail addressed to the user has been received by the server 5, then the server 5 will send a message to the client 1 indicating this fact. The client 1 will then notify the user that e-mail is waiting by a message displayed on the television set 12 or by an LED (light emitting diode) built into the housing of WebTV™ box 10.

Thus, a method and apparatus have been described for providing proxying and transcoding of documents in a network. The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

17

What is claimed and desired to be secured by United States Letters Patent is:

1. In a proxying server coupled to a client and to a remote server, the proxying server operating as a proxy on behalf of the client for accessing the remote server, a method of retrieving and transcoding a document requested by the client, the method comprising the steps of:

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document in order to perform at least one function selected from the following functions:

matching decompression requirements at the client; converting the document into a format compatible for the client;

reducing latency experienced by the client; and altering the document to fit into smaller memory space; and

transmitting the document to the client.

2. A method according to claim 1, wherein the request includes a URL (Uniform Resource Locator).

3. A method according to claim 1, wherein the step of transcoding is performed while the step of retrieving the document is performed.

4. A method according to claim 1, wherein the step of transcoding comprises the step of reading at least a portion of the document from a proxy cache located in the proxying server.

5. A method according to claim 1, wherein the step of transmitting the document to the client is performed prior to performing the step of transcoding, at the proxying server, the data in the document.

6. A method according to claim 1, wherein the step of transmitting the document to the client comprises the step of transmitting the transcoded document to the client, wherein the step of transmitting the document to the client is performed after the step of transcoding, at the proxying server, the data in the document.

7. A method according to claim 1, wherein the document includes a link to another document, the link including a retrieval address, and wherein the step of transcoding at the proxying server the data in the document comprises the step of updating the link

8. A method according to claim 7, wherein the other document is an image, and wherein the step of updating the link comprises the step of adding information to the document indicating the size of the image.

9. A method according to claim 8, wherein the other document is inaccessible to the proxying server, and wherein the step of updating the link comprises the step of removing the link.

10. A method according to claim 7, wherein the other document has been relocated from the retrieval address to a redirect address, and wherein the step of updating the link comprises the step of updating the link to correspond to the redirect address.

11. A method according to claim 1, wherein the client includes a television display, wherein the document references an image, and wherein the step of transcoding, at the proxying server, the data in the document is conducted in order to perform at least the function of converting the document into a compatible format for the client, and comprises the step of revising the data such that the image is sized for display on the television display.

18

12. A method according to claim 1, wherein the document references an image having a first image format, and wherein the step of transcoding, at the proxying server, the document is conducted in order to perform at least the function of converting the document into a compatible format for the client, and includes the step of converting the first image format to a second image format.

13. A method according to claim 1, further comprising the steps of:

identifying an image referenced by the document;

determining whether the image has been previously retrieved by the proxying server; and

if the image has been previously retrieved by the proxying server, accessing information stored in the proxying server indicating the size of the image;

wherein the step of transcoding, at the proxying server, the data in the document is conducted in order to perform at least the function of reducing latency experienced by the client, and comprises the step of using the information indicating the size of the image to revise the data of the document to allow the document to be partially displayed by the client before the image is received by the client.

14. A method according to claim 13, wherein the step of transcoding, at the proxying server, the data in the document comprises the following step:

if the image has been previously retrieved by the proxying server, transcoding at the proxying server the data in the document without the image but providing a space for the image to be later inserted; and

wherein the step of transmitting the document to the client comprises the following steps:

transmitting the document to the client without the image but providing the space for the image to be later inserted; and

after transmitting the document to the client without the image, transmitting the image to the client.

15. In a proxying server coupled to a client and to a remote server, the proxying server operating as a proxy on behalf of the client for accessing the remote server, a method of retrieving and transcoding a document requested by the client, the method comprising the steps of:

providing a persistent database at the proxying server, the persistent database including information relating to the document;

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document using the information included in the persistent database in order to perform at least one function selected from the following functions:

matching decompression requirements at the client; converting the document into a format compatible for the client;

reducing latency experienced by the client; and altering the document to fit into smaller memory space; and

transmitting the document to the client.

16. A method according to claim 15, further comprising the step of storing in the persistent database validity information corresponding to the document.

19

17. A method according to claim 16, wherein the step of retrieving the document comprises the following steps:

retrieving the document from the persistent database if the validity information corresponding to the document indicates that the document is valid; and

retrieving the document from the remote server if the validity information corresponding to the document indicates that the document is not valid.

18. A method according to claim 16, wherein the step of storing in the persistent database validity information corresponding to the document comprises the step of observing a rate of change of the document.

19. A method according to claim 18, wherein the step of observing a frequency of change of the document comprises the step of periodically retrieving the document, wherein successive retrievals of the document are separated in time by at least a time interval.

20. A method according to claim 19, wherein the step of periodically retrieving the document comprises the following steps:

retrieving the document at a first time;

storing the document retrieved at the first time in a memory;

retrieving the document at a second time, the second time being at least the time interval after the first time;

determining whether the document retrieved at the second time has changed compared to the document retrieved at the first time;

if the document retrieved at the second time is different than the document retrieved at the first time, replacing the document retrieved at the first time with the document retrieved at the second time in the memory; and

if the document retrieved at the second time is the same as the document retrieved at the first time, extending the time interval.

21. A method according to claim 15, further comprising the step of storing in the persistent database performance information relating to performance of the remote server when accessing the document.

22. A method according to claim 21, wherein the performance information comprises a latency value.

23. A method according to claim 15, further comprising the step of storing in the persistent database information for optimizing memory usage by the client.

24. A method according to claim 15, wherein the step of retrieving the document comprises the following steps:

determining whether a redirect address corresponding to the document is stored in the persistent database;

if the redirect address corresponding to the document is stored in the persistent database, accessing the remote server using the redirect address, the remote server corresponding to the redirect address; and

if the redirect address corresponding to the document is not stored in the persistent database, the method further comprises the following steps:

accessing a previous remote server at which the document previously resided;

obtaining the redirect address from the previous remote server;

storing the redirect address in the persistent database; and

20

accessing the remote server using the redirect address.

25. A computer program product for implementing, in a proxying server coupled to a client and to a remote server, a method of retrieving and transcoding a document requested by the client, the computer program product comprising a computer-readable medium carrying computer-executable instructions for causing the proxying server to perform acts included in the method, said acts comprising:

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document in order to perform at least one function selected from the following functions:

matching decompression requirements at the client;

converting the document into a format compatible for the client;

reducing latency experienced by the client; and
altering the document to fit into smaller memory space; and

transmitting the document to the client.

26. A computer program product according to claim 25, wherein the computer-executable instructions comprise a plurality of program code means for transcoding at least a portion of the document, including:

program code means for transcoding at least a portion of the document so as to match decompression requirements at the client;

program code means for transcoding at least a portion of the document so as to convert the document into a format compatible with the client;

program code means for transcoding at least a portion of the document so as to reduce latency experienced by the client; and

program code means for transcoding at least a portion of the document so as to alter the document to fit into smaller memory space.

27. A computer program product according to claim 26, wherein the act of transcoding comprises selecting at least one of said plurality of program code means for transcoding for use in the act of transcoding.

28. A computer program product according to claim 25, wherein the act of transcoding is performed while the act of retrieving the document is performed.

29. A computer program product according to claim 25, wherein the act of transcoding comprises reading at least a portion of the document from a proxy cache located in the proxying server.

30. A computer program product according to claim 25, wherein the act of transmitting the document to the client is performed prior to performing the act of transcoding.

31. A computer program product according to claim 25, wherein the act of transmitting the document to the client comprises the act of transmitting the transcoded document to the client, wherein the act of transmitting the document to the client is performed after the act of transcoding.

* * * * *



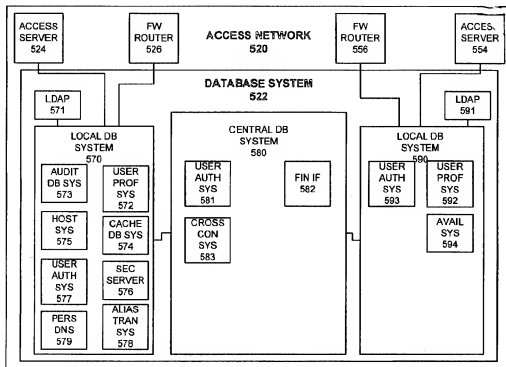
US006697806B1

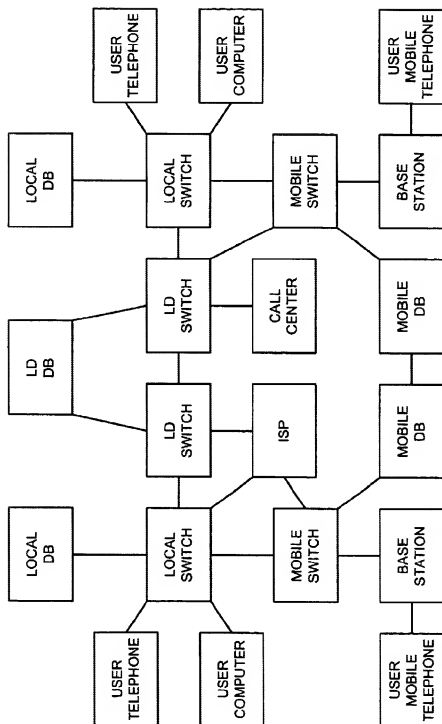
**(12) United States Patent
Cook****(10) Patent No.: US 6,697,806 B1
(45) Date of Patent: Feb. 24, 2004****(54) ACCESS NETWORK AUTHORIZATION****(75) Inventor: Fred S. Cook, Olathe, KS (US)****(73) Assignee: Sprint Communications Company,
L.P., Overland Park, KS (US)****(*) Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.**(21) Appl. No.: 09/574,978****(22) Filed: May 19, 2000****Related U.S. Application Data****(63)** Continuation of application No. 09/556,276, filed on Apr. 24, 2000.**(51) Int. Cl.⁷ G06F 17/30****(52) U.S. Cl. 707/10; 707/3; 707/9;
709/227; 709/203; 709/219****(58) Field of Search 707/1-3, 9-10,
707/4, 104.1; 709/223-227, 219, 203****(56) References Cited****U.S. PATENT DOCUMENTS**5,884,312 A * 3/1999 Dustan et al. 707/10
6,151,601 A * 11/2000 Papierniak et al. 707/1

* cited by examiner

Primary Examiner—Jean R. Homere
Assistant Examiner—Mohammad Ali**(57) ABSTRACT**

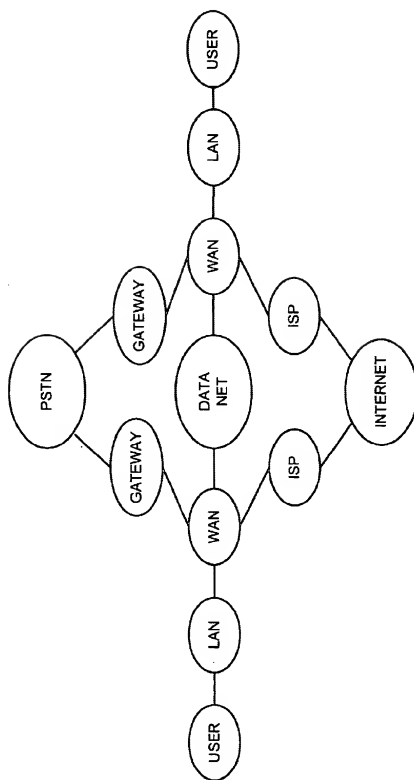
An access communication system provides access between a user system and a plurality of communication networks. The plurality of communication networks provide services to a user in the user system. An access communication system includes a local database system and an access server that is connected to the user system and the plurality of communication networks. The local database system receives a user logon. The local database system then processes the user logon to determine if the user is allowed access to the access communication system based on a local database system. The local database system then provides access to the access communication system to the user in response to the determination that the user is allowed access based on the local database system. The local database system then generates an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system. The local database system receives and processes an authorization response indicating whether the user is allowed to use the access system from the second database system. The local database system then provides access to the access communication system to the user in response to the authorization response that allows the user to use the access communication system.

24 Claims, 63 Drawing Sheets



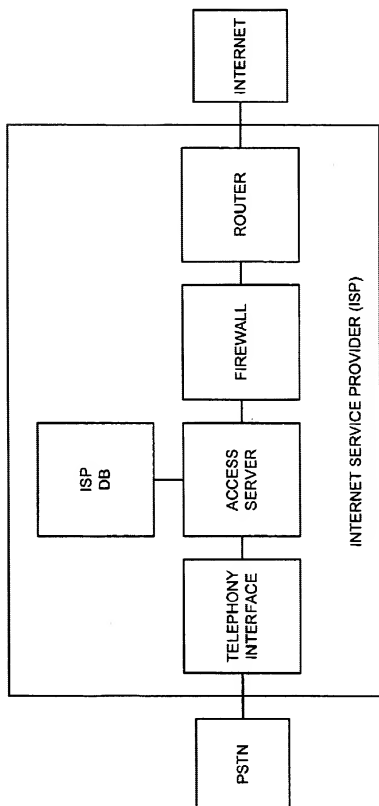
PRIOR ART

FIG. 1



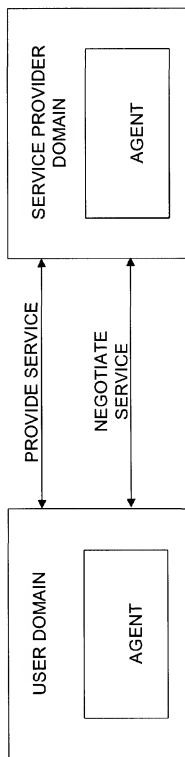
PRIOR ART

FIG. 2



PRIOR ART

FIG. 3



PRIOR ART

FIG. 4

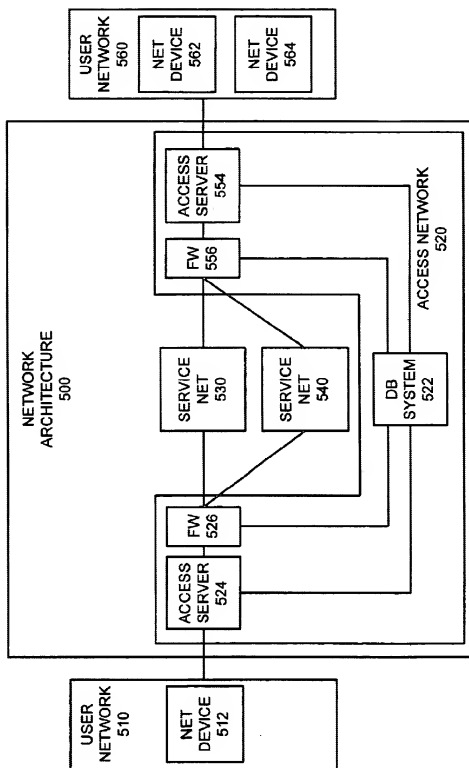


FIG. 5

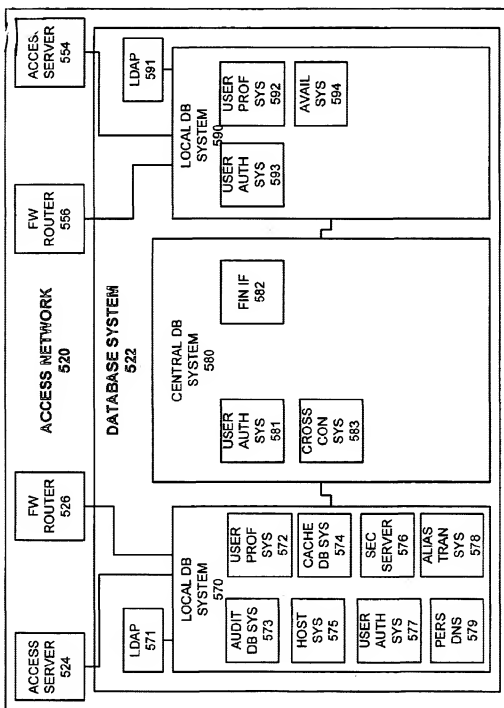


FIG. 6

USER ID	PASSWORD	NAME	ACCOUNT NUMBER	SERVICES	ADDRESS	BILLING CODE	CLASS	GROUP	SHELL	MACROS

FIG. 7

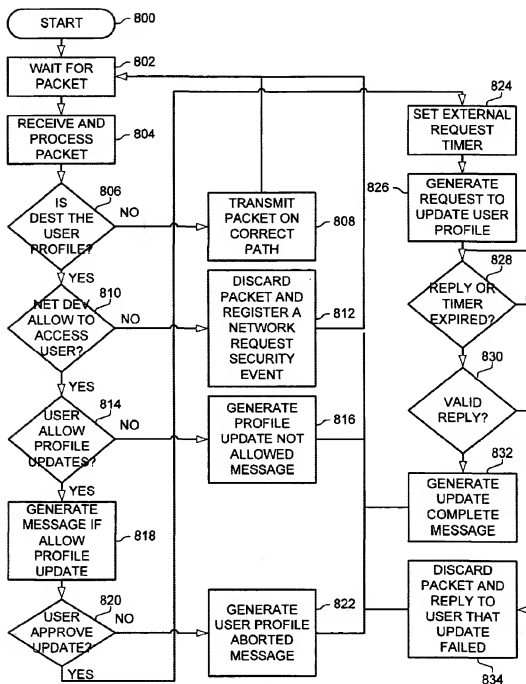


FIG. 8

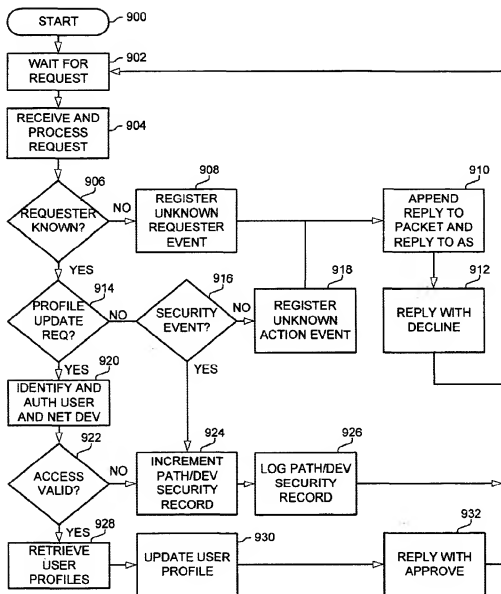


FIG. 9

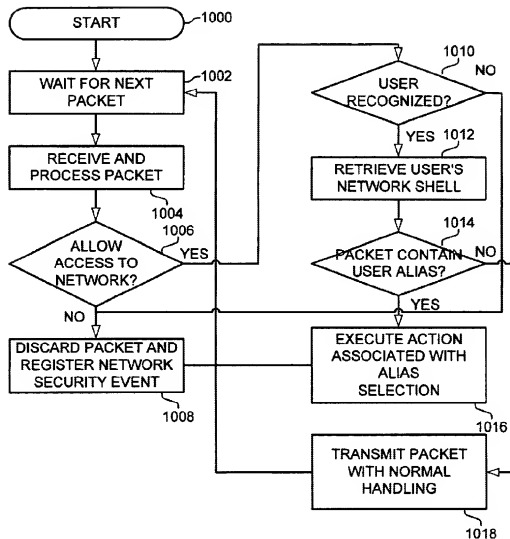


FIG. 10

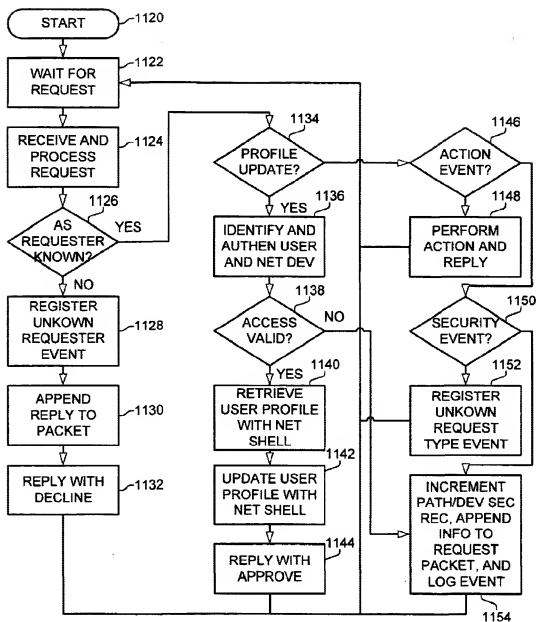
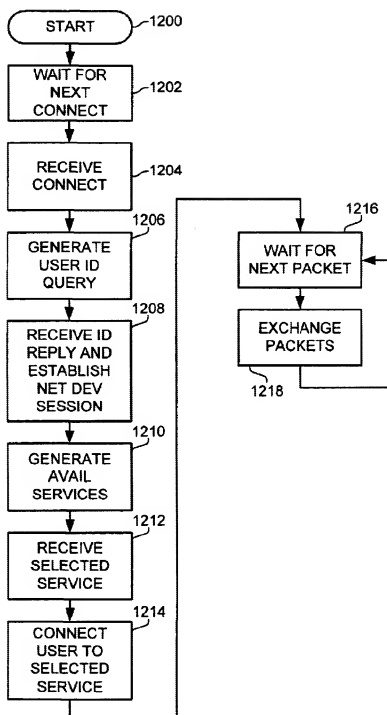


FIG. 11

**FIG. 12**

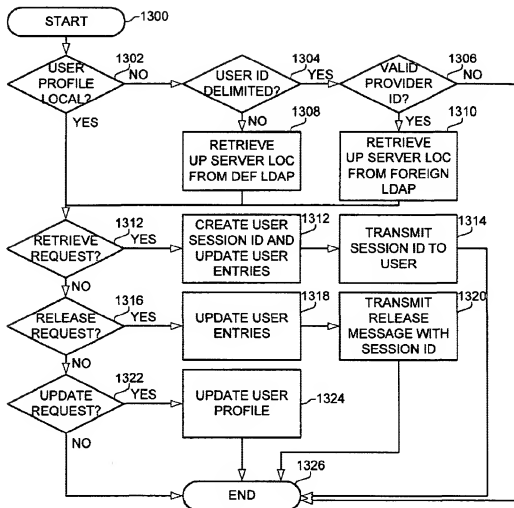


FIG. 13

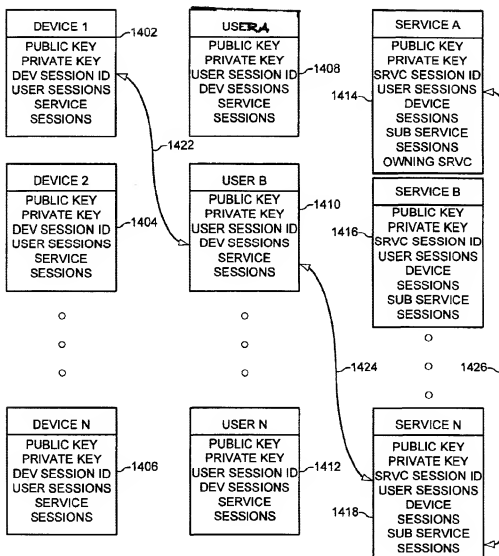
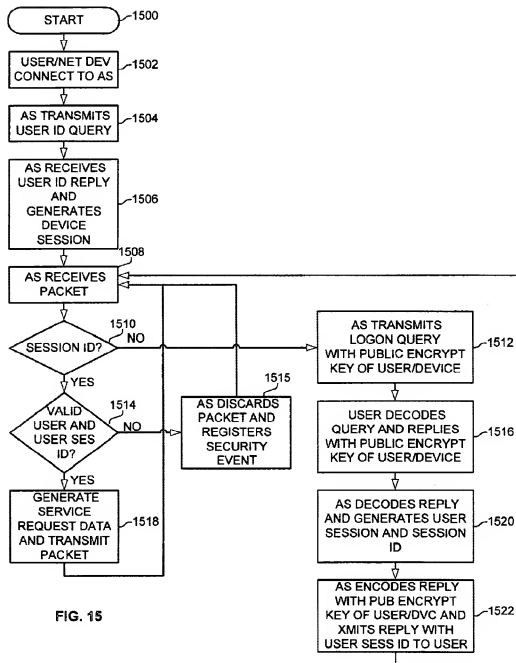


FIG. 14



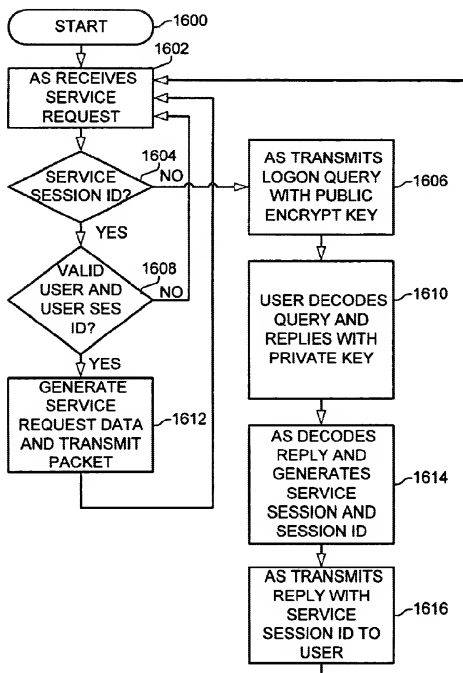


FIG. 16

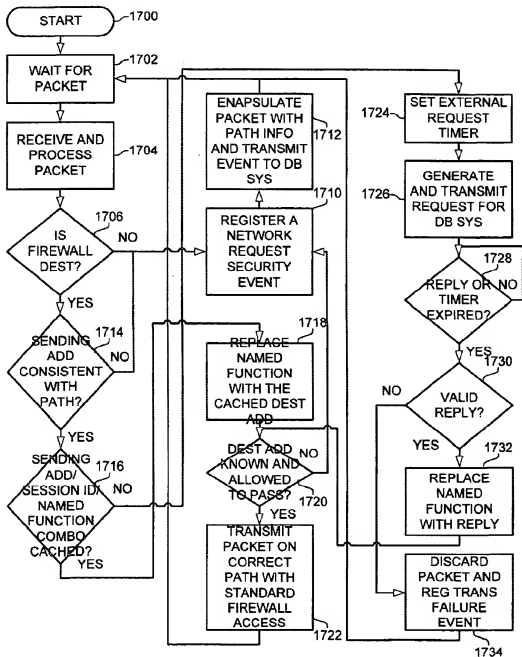


FIG. 17

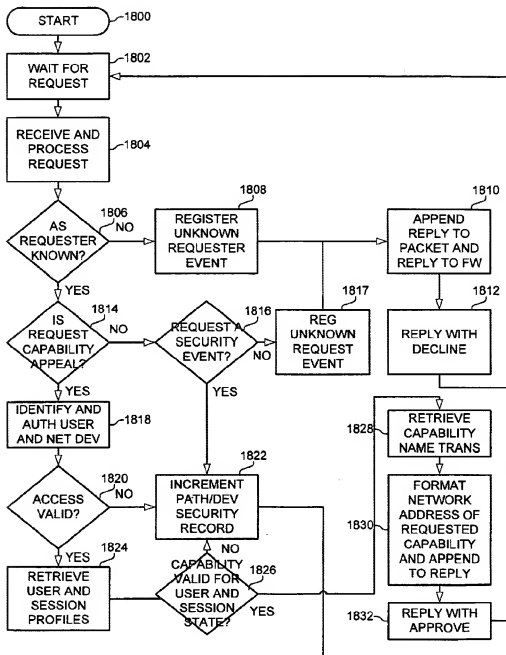


FIG. 18

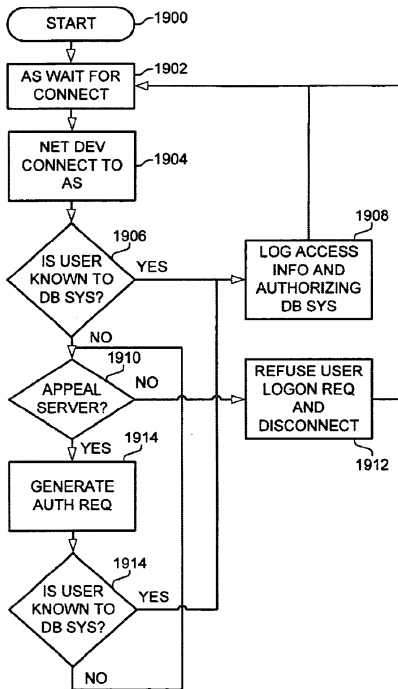


FIG. 19

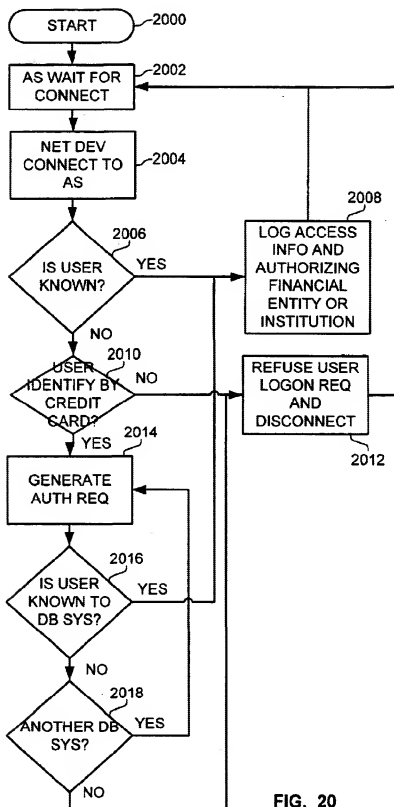


FIG. 20

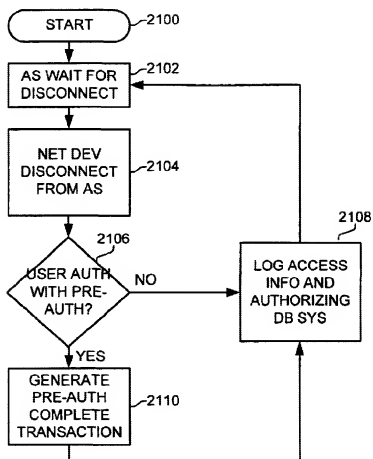


FIG. 21

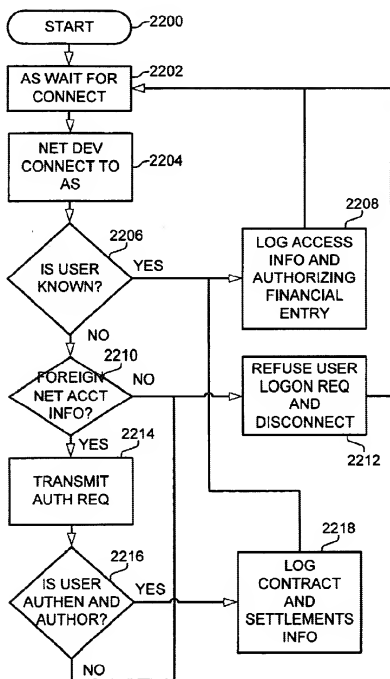
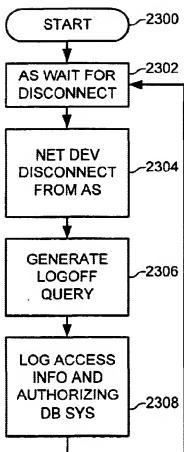


FIG. 22

**FIG. 23**

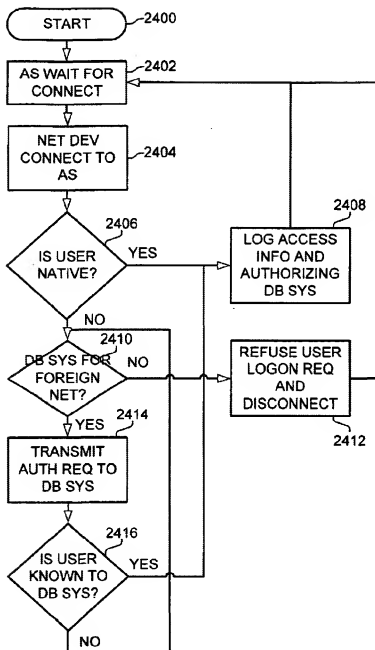


FIG. 24

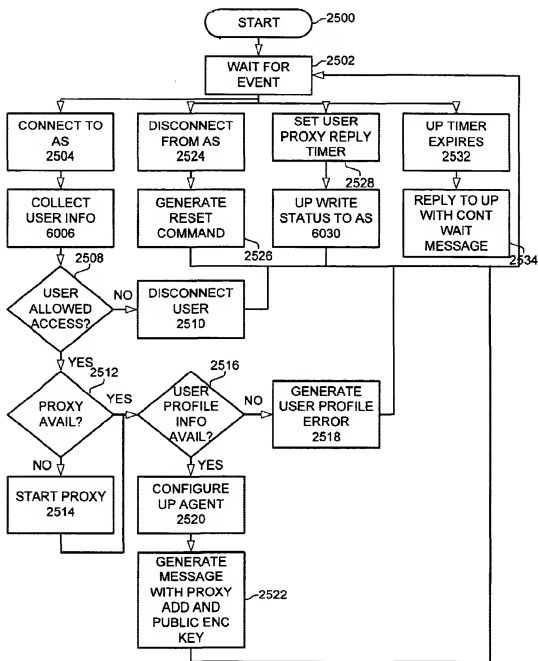


FIG. 25

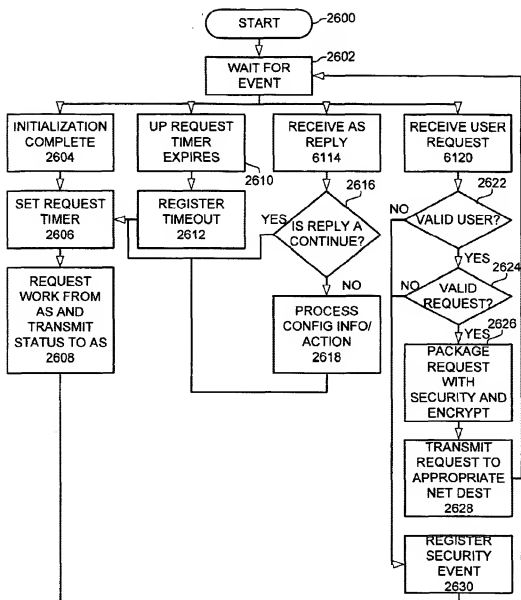


FIG. 26

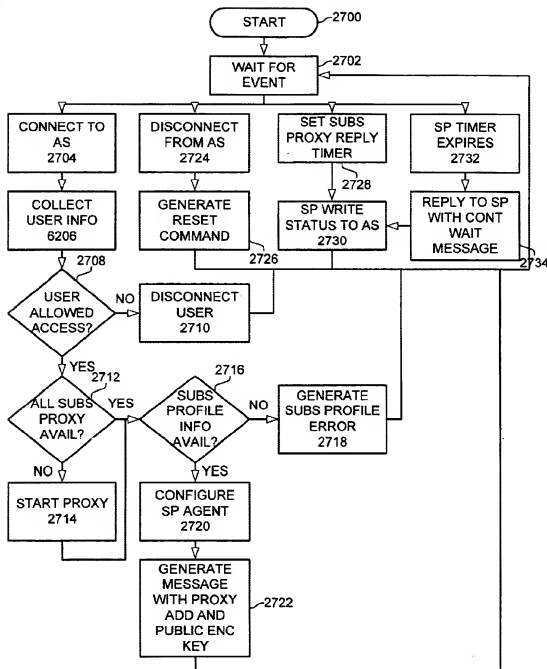


FIG. 27

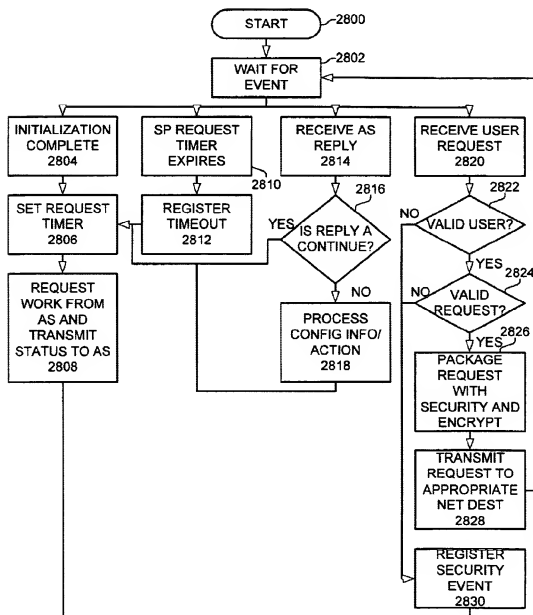


FIG. 28

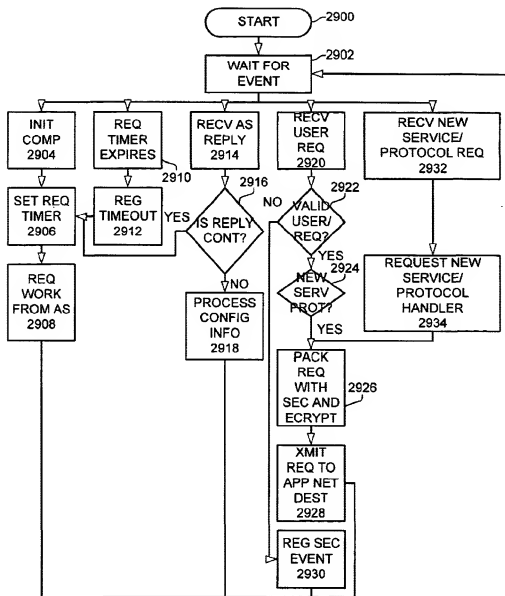


FIG. 29

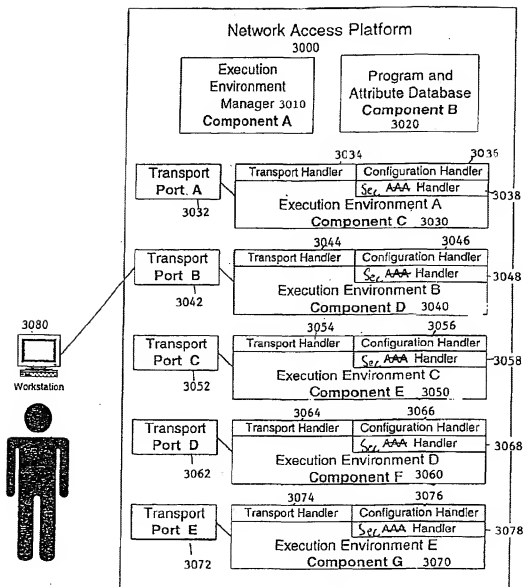


FIGURE 30

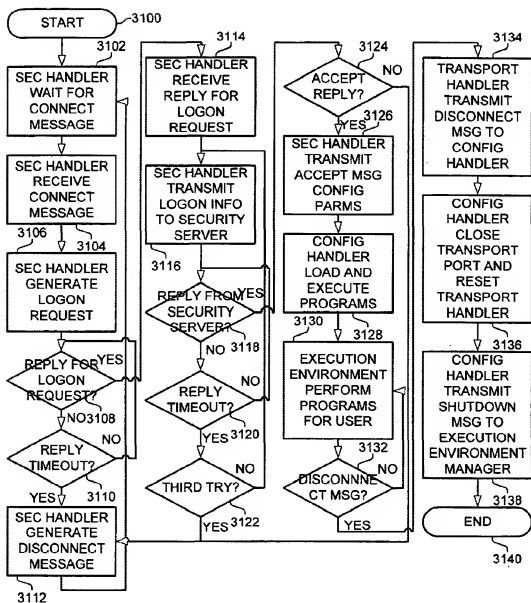


FIG. 31

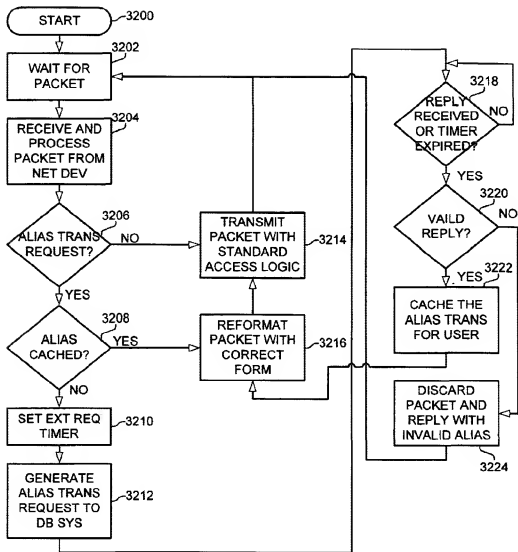


FIG. 32

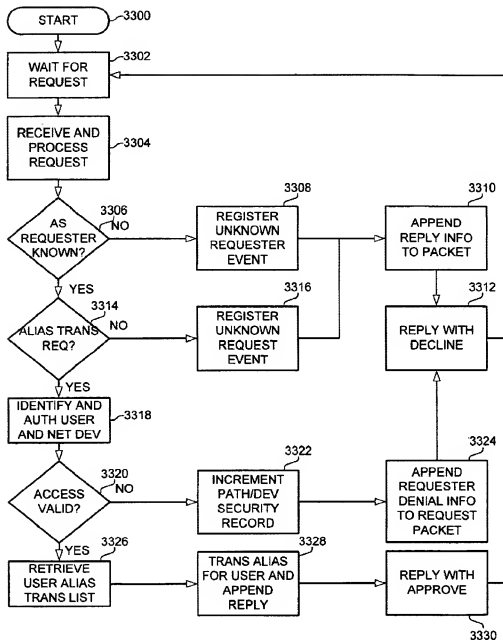


FIG. 33

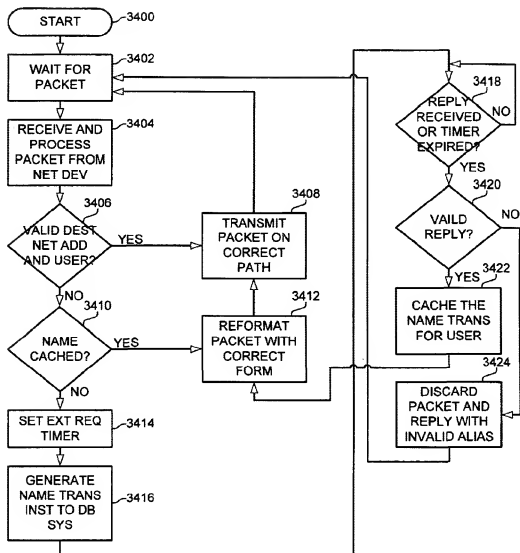


FIG. 34

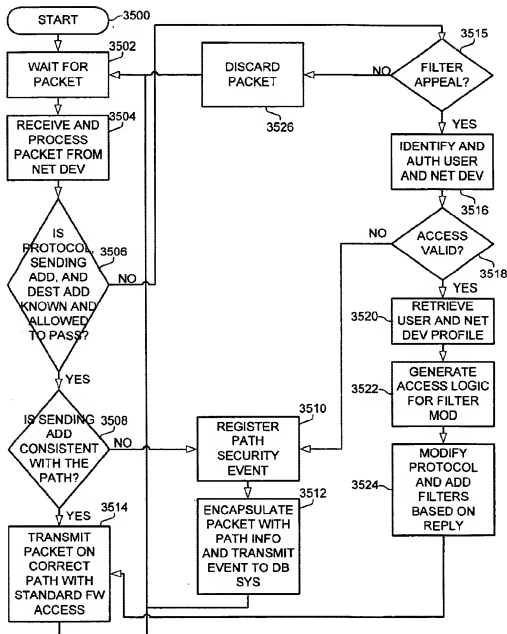


FIG. 35

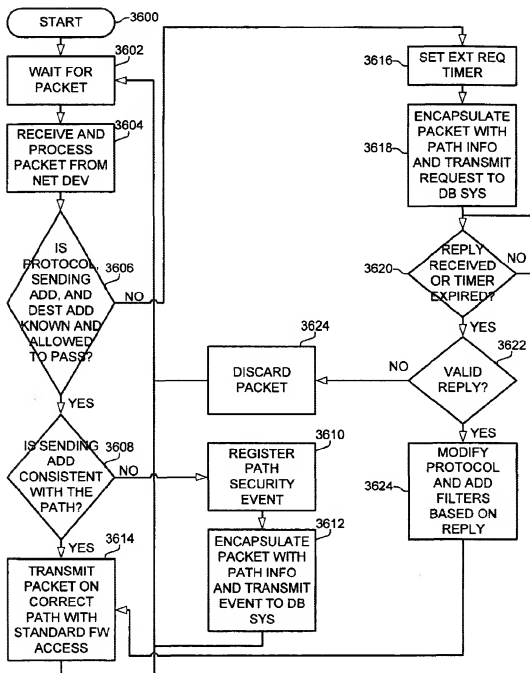


FIG. 36

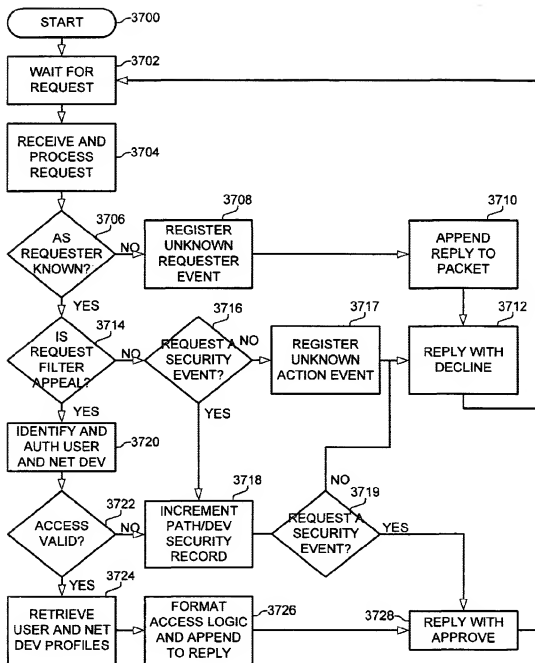


FIG. 37

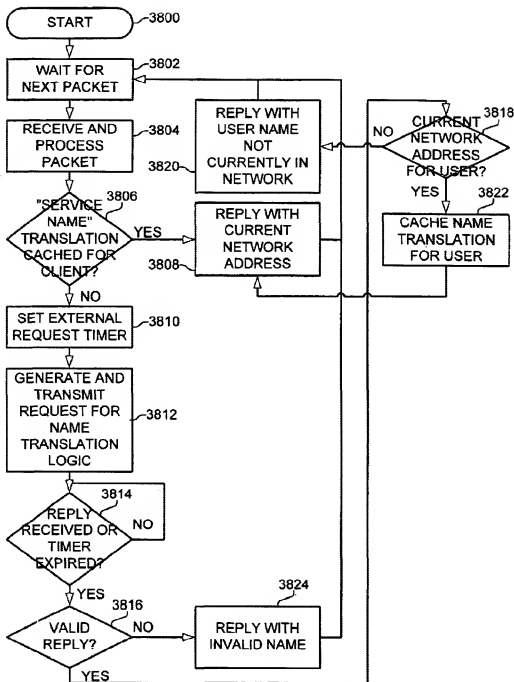


FIG. 38

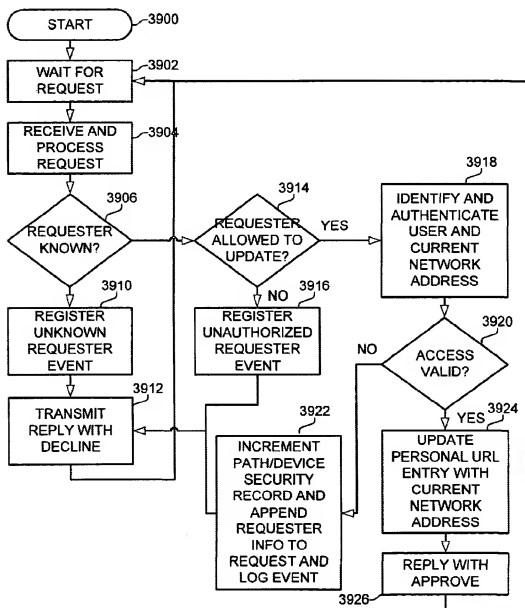


FIG. 39

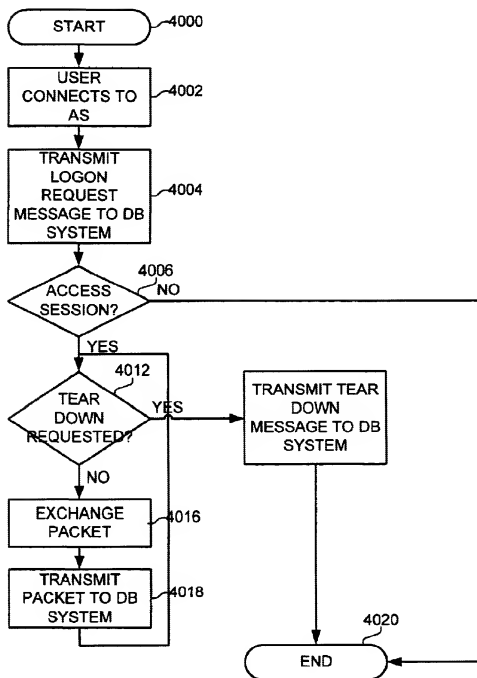


FIG. 40

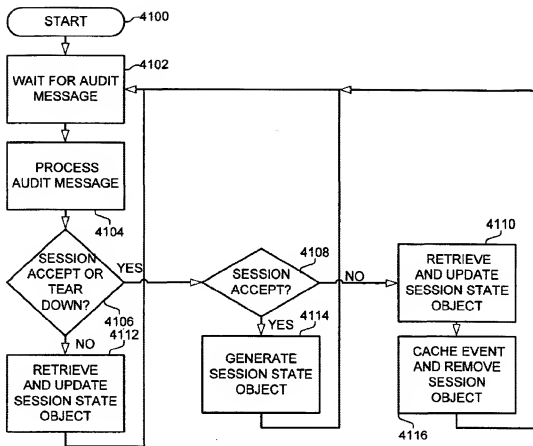


FIG. 41

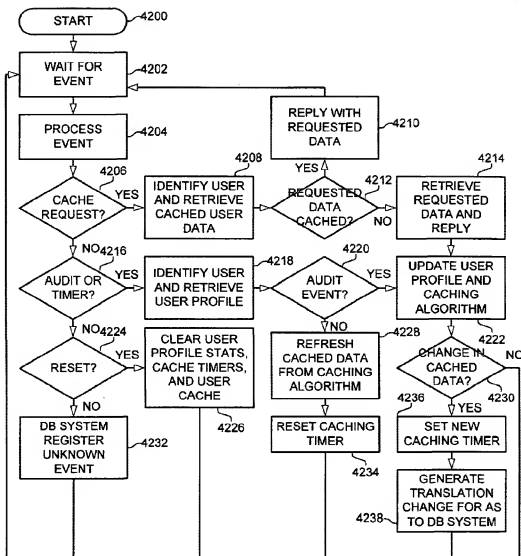


FIG. 42

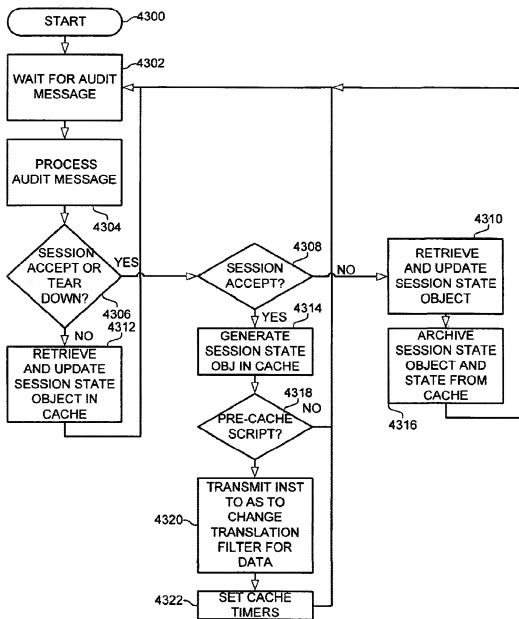


FIG. 43

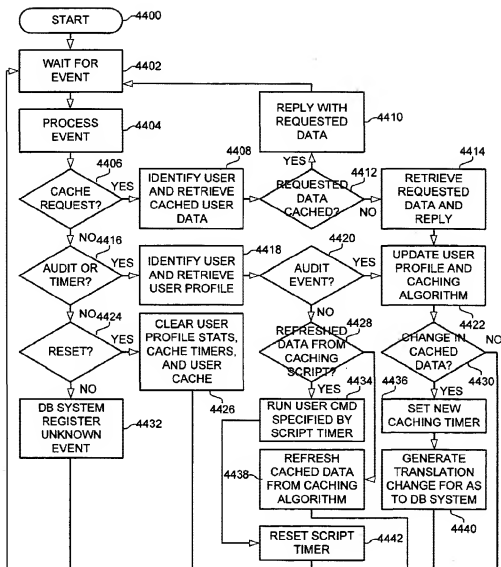


FIG. 44

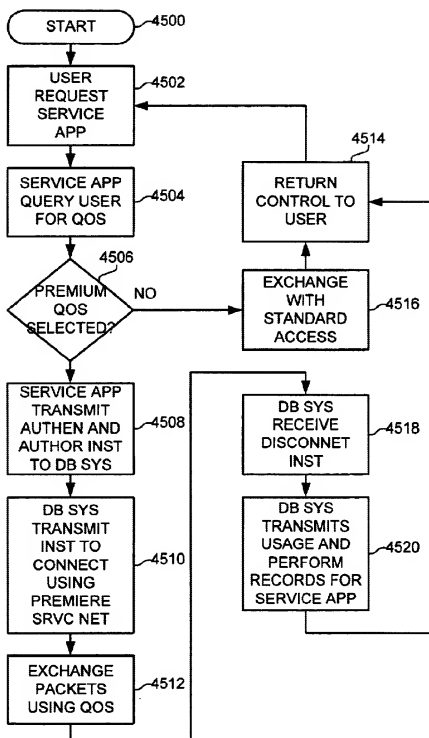


FIG. 45

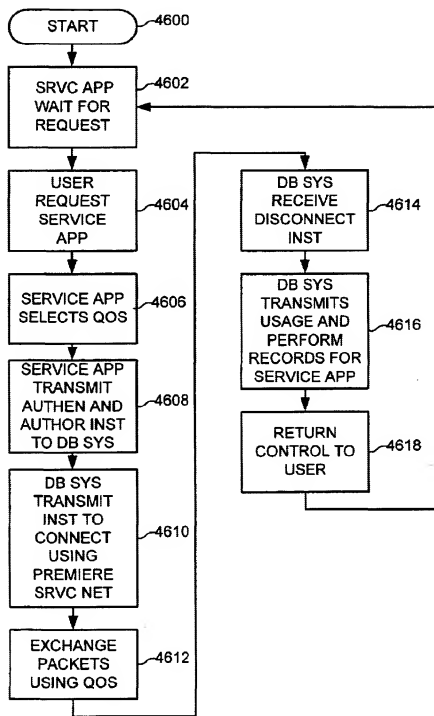


FIG. 46

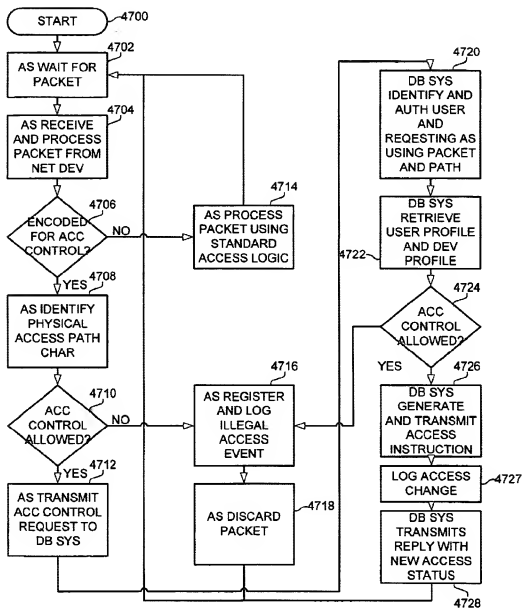


FIG. 47

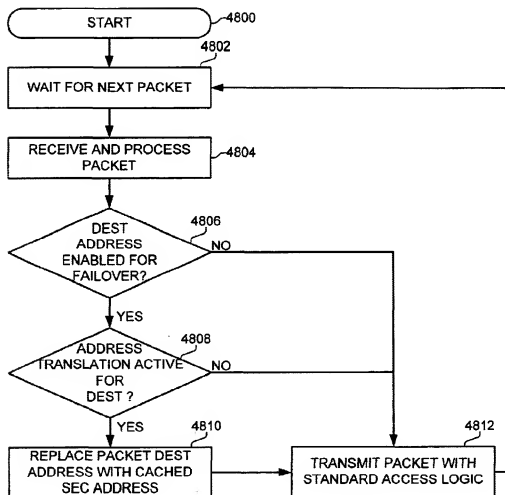


FIG. 48

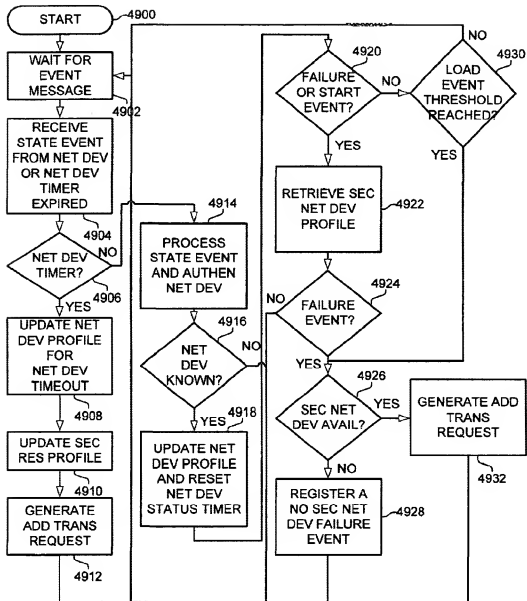


FIG. 49

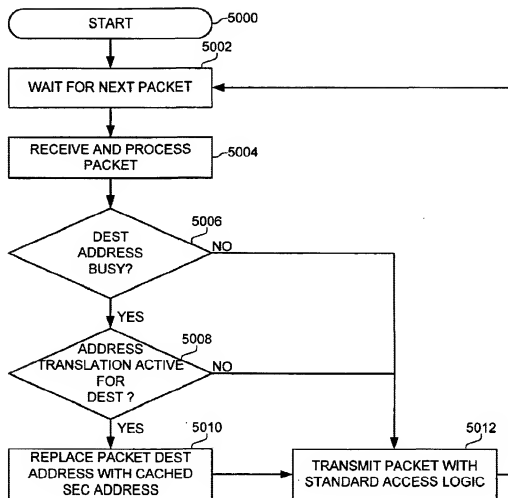


FIG. 50

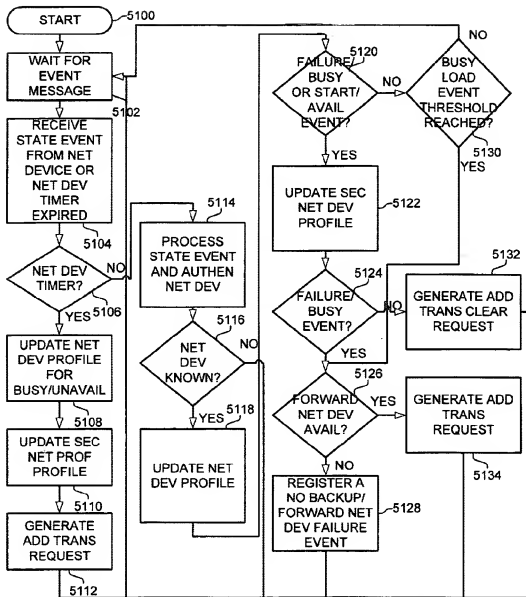


FIG. 51

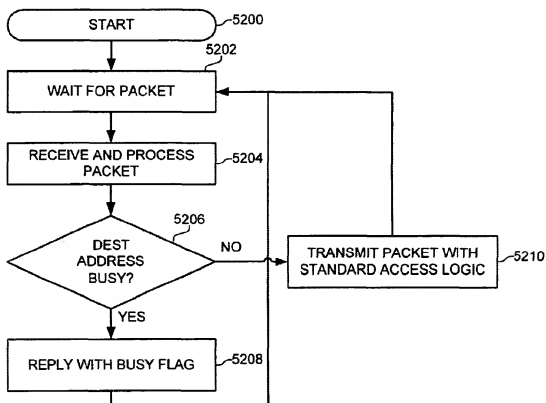


FIG. 52

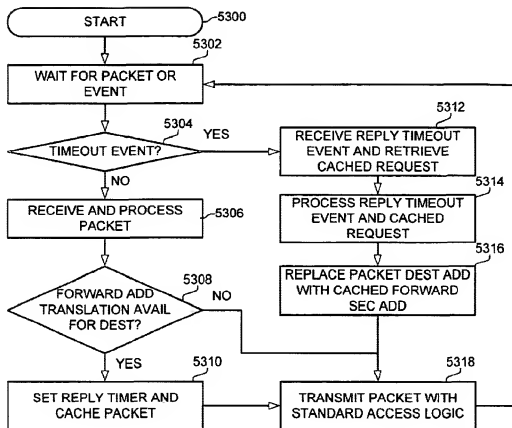


FIG. 53

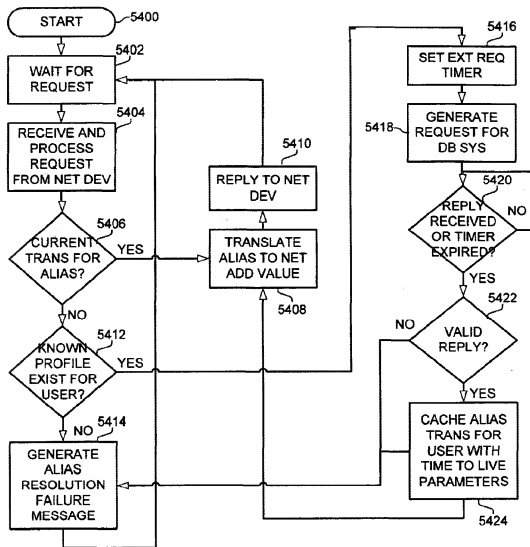


FIG. 54

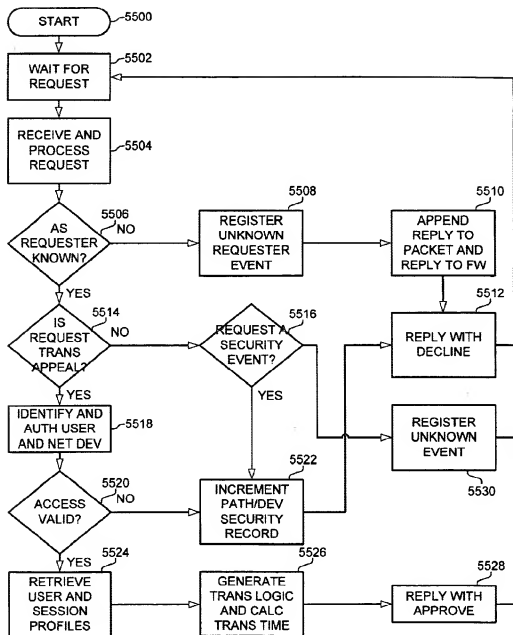


FIG. 55

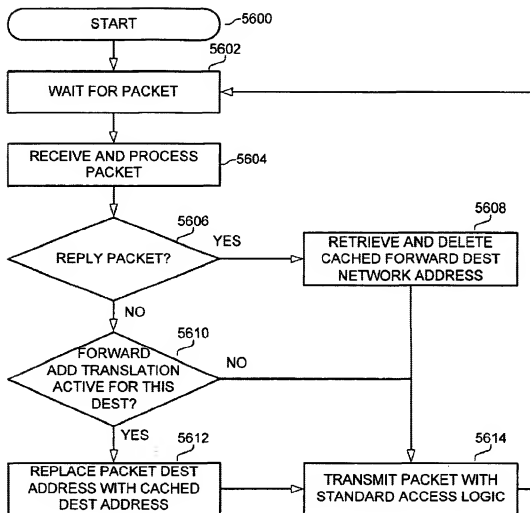


FIG. 56

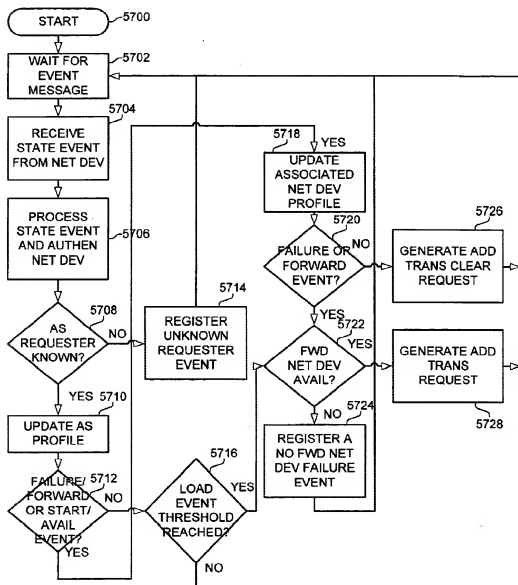


FIG. 57

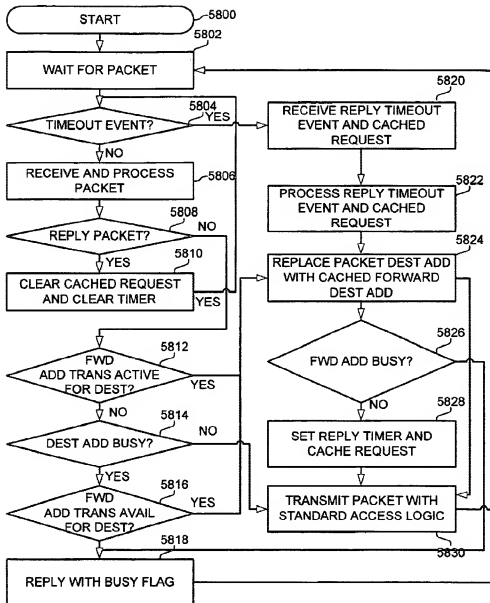


FIG. 58

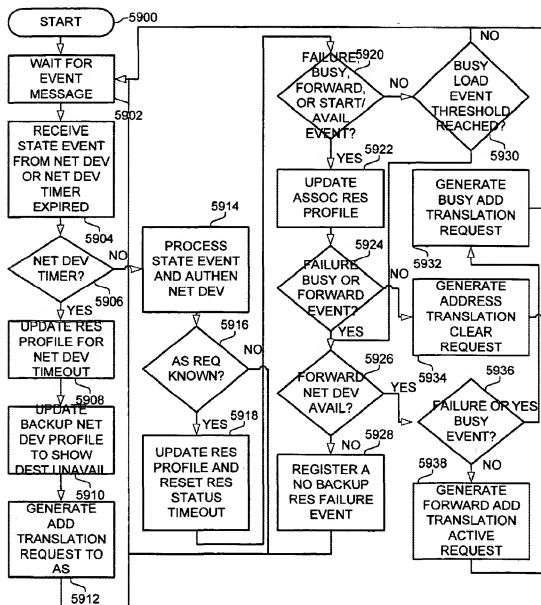
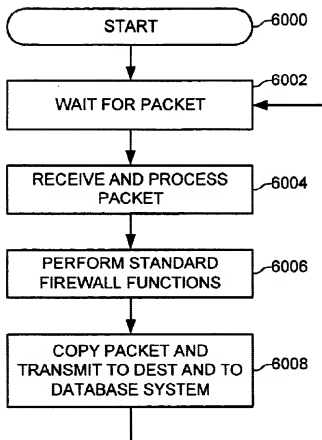


FIG. 59

**FIG. 60**

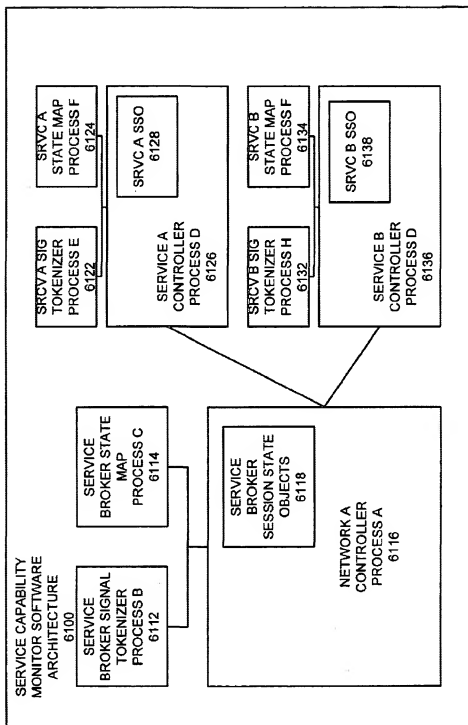
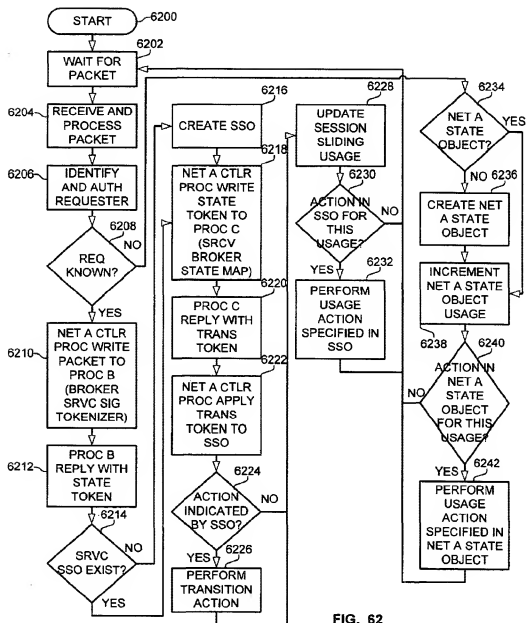


FIG. 61



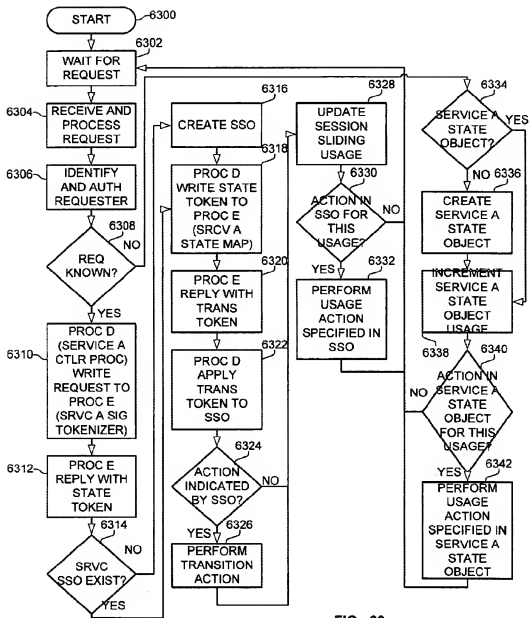


FIG. 63

ACCESS NETWORK AUTHORIZATION

RELATED APPLICATIONS

This application is a continuation of prior application Ser. No. 09/556,276, entitled "Access Communication System", filed Apr. 24, 2000, currently pending, and incorporated by reference into this application.

FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

Not applicable

MICROFICHE APPENDIX

Not applicable

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention is related to the field of communication networks, and in particular, to an access communication system that provides access to multiple service provider systems. More particularly, this invention relates to a system that authorizes access to use the access communication system.

2. Description of the Prior Art

Communication networks have seen dramatic development over the past several years so that today, there are multiple diverse communication networks providing services. The current technical challenge is to develop interfaces between the networks to provide seamless service across multiple networks. Unfortunately, today's interfaces lack the ability to offer the user with easy access to services from multiple systems. These interfaces do not customize operations for the user.

FIG. 1 illustrates the conventional public telephone network. User telephones and computers are connected to local switches. The local switches are coupled to a local database. The user places calls through the local switch. The local switch processes the called number to provide an end-point connection or access to other networks. The local switch may connect the call to another telephone in the local calling area. In a Local Number Portability (LNP) situation, the local switch exchanges information with the local database to obtain the appropriate routing number for a ported call. The local switch may also connect the call to an Internet Service Provider. If a Digital Subscriber Line (DSL) is used, DSL equipment may be used to bypass the local switch. The local switch may also connect the call to a long distance switch. To provide access to the long distance switch, the local switch first exchanges information with the local database. The local database identifies the long distance network for either the user or the dialed number.

The long distance switch processes the called number to route the call to another system or network. Prior to routing, the long distance switch validates the call by checking the caller's number. The long distance switch may also exchange information with the long distance database to provide special call-handling. One example is a calling card call, where the long distance database validates an account number for billing the call. Another example is a toll-free call, where the long distance database processes external information customized by the called party to route the call. Toll-free routing information includes items such as time and date, caller location, and call center status. Many long distance calls are simply routed through the long distance

network to another local network for call completion. Other calls are routed from the long distance network to a call center. Call centers offer a concentration of call-handling capabilities for operations, such as order entry, customer service, and promotions. Call centers include automatic call distribution equipment to route calls to the appropriate destination within the call center.

Both local and long distance networks exchange calls with mobile switches. The mobile switches are connected to base stations that communicate over the air with wireless telephones. When a mobile user places a call, the mobile switches exchange information with a mobile database to validate the mobile caller. The call is then routed to another mobile caller, the telephone network, or to an ISP. When a mobile caller moves around, their wireless phone logs-in with the physically proximate mobile network, and that mobile network updates the mobile user's location in the mobile databases. When a call is placed to that mobile user, their home mobile switch obtains a routing number from the mobile databases to route the call to the mobile network currently in communication with the mobile user.

FIG. 2 illustrates a conventional data network that transfers packets of user data to a destination based on address information carried in the packets. Users are connected to Local Area Networks (LANs) that are connected to Wide Area Networks (WANs). A common LAN is an Ethernet system. A common WAN is an intranet. WANs are interconnected by data networks, such as IP, TI, frame relay, or Asynchronous Transfer Mode (ATM). WANs are connected to the Internet through ISPs. WANs are connected to the public telephone network through telephony gateways. A common telephony gateway is a Private Branch Exchange (PBX).

FIG. 3 illustrates a conventional ISP. The public telephone network is coupled to a telephony interface that converts between telephony analog and digital protocols and the Internet Protocol (IP). Some telephony interfaces also handle DSL traffic that may already use IP. The telephony interface transfers IP traffic through an access server and firewall to a router. Some ISPs combine the firewall and the access server into one system. Also, the position of the firewall may vary, and traffic shapers may be present. The router exchanges IP traffic with the Internet.

In operation, the user calls the ISP over the telephone network and logs-in at the access server. The access server collects and forwards the user name and password to the ISP database. The ISP database validates the user name and password and returns an IP address to the access server. The IP address is for the user's terminal connection. Using the IP address, the user may communicate through the firewall to the router for transmissions to an IP address. The user now has Internet access through the router and exchanges packets with various Internet servers.

IP addresses are referred to as network addresses and include a network ID and a host ID. Network IDs are unique across the Internet and host IDs are unique within a given network. IP addresses are lengthy numerical codes, so to simplify things for the user, service addresses are available that are easier to remember. The service addresses are often the name of the business followed by ".com". Domain Name Service (DNS) is hosted by servers on the Internet and translate between service addresses and network addresses. The browser in the user computer accesses the DNS to obtain the desired network address.

FIG. 4 illustrates conventional network access. A current proposal for communication network access is provided by

3

the Telecommunication Information Network Architecture Consortium (TINA-C). TINA-C proposes the use of agents in the user domain and the service provider domain. The service provider domain could be a telephone network, data network, or ISP. The agents negotiate access service rights. Once the service is negotiated, the user receives the service from the service provider network during a service session. Unfortunately, the access session occurs between the user domain and a particular service provider domain. At present, the service provider domain provides limited access capability beyond simply handing off communications to another network based on a called number or network address. As a result, the ability to customize services for a particular user across multiple service providers is inadequate.

SUMMARY OF THE INVENTION

The inventions solve the above problems by providing access between a user system and a plurality of communication networks. The plurality of communication networks provide services to a user in the user system. An access communication system includes a database system and an access server that is connected to the user system and the plurality of communication networks.

In one aspect of the inventions for user access profile inheritance, the database system receives an update request from the access server to update a user access profile through inheritance. The database system then processes the update request to inherit user profile information from a user profile data structure. The database system updates the user access profile with the user profile information.

In another aspect of the inventions for network shells, the access server receives an alias selection from a user for a network shell that includes alias selections associated with actions. The access server then processes the alias selection to execute an action associated with the alias selection.

In another aspect of the inventions for service based directory, the access server transmits a list of services to a user system. The access server then receives a selection from the list of services. The access server processes the selection to generate an instruction to provide the service related to the selection.

In another aspect of the inventions for user access profile mobility, the database system receives user information. The database system then processes the user information to determine if a user access profile is local within a local database system. The database system generates and transmits a request to retrieve a user access profile from a second database system external to the local database system in response to the determination that the user access profile is not local.

In another aspect of the inventions for service, user, and device sessions, the access communication system establishes a connection between a network device and the access server. The access communication system then generates a device session including a device session ID based on the network device. The access communication system generates and transmits a login query for the network device. The access communication system receives and processes a login reply from the network device to generate a user session including a user session ID based on the user. The access communication system receives and processes a request for the service to generate a service session including a service session ID based on the service. The service may generate and transmit a login query for the user. The access communication system links the device session, user session, and the service session using the device session ID, the user session ID, and the service session ID.

4

In another aspect of the inventions for service capability firewall, the access server receives information including a named function request for a service provider. The access server processes the information to check if the named function request is valid for the service provider and the service. If valid, the access server determines if a private destination address exists for the named function request. The access server replaces the named function request with the private destination address in response to the determination that the private destination address exists for the named function request. The access server then transmits the information with the private destination address to the service provider.

In another aspect of the inventions for prepaid access and bank card access, the database system receives information identifying a billing code for a user. The database system then processes the billing code to determine if the user is allowed to use the access system. The database system provides access to the access system in response to the determination that the user is allowed to use the access system.

In another aspect of the inventions for global authentication and access card, the database system receives a user login. The database system then processes the user login to determine if the user is allowed access to the access communication system based on a local database system. The database system then provides access to the access communication system to the user in response to the determination that the user is allowed access based on the local database system. The database system then generates an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system. The database system receives and processes an authorization response indicating whether the user is allowed to use the access system from the second database system. The database system then provides access to the access communication system to the user in response to the authorization response that allows the user to use the access communication system.

In another aspect of the inventions for user based proxy and subscriber based proxy, the database system includes a user proxy. The user proxy receives a request for the service from the user system. The user proxy then transmits the request for the service to a service provider. The user proxy exchanges user information between the user system and the service provider.

In another aspect of the inventions for dynamic proxy, the database system includes a proxy. The proxy receives a service/protocol request for a new service or protocol. The proxy processes the service/protocol request to generate a handler request to obtain a handler for the new service or protocol. The proxy then receives and executes the handler for the new service or protocol.

In another aspect of the inventions for access execution environment, the database system receives and processes a login reply into an access execution environment for a user. The database system retrieves programs for the user into the access execution environment. The database system executes the programs for the user in the access execution environment.

In another aspect of the inventions for domain name scoping and inband domain name service lookup, the access server receives information including an alias from the user system. The access server determines if the alias exists in a cache including aliases and alias translations for the user.

5

The access server changes the information based on the cached alias translation.

In another aspect of the inventions for inline access service triggering, the access server receives information. The access server then processes the information to determine if the information is allowed to pass. The access server changes access logic based on the information in response to the determination that the information is not allowed to pass. The access server changes the filters of the access server based on the information in response to the determination that the information is not allowed to pass.

In another aspect of the inventions for access service triggering, the access server receives information. The access server processes the information to determine if the information is allowed to pass. The access server then generates a request from a database system in response to the determination that the information is not allowed to pass. The access server receives a reply including access logic from the database system. The access server changes filters of the access server based on the access logic.

In another aspect of the inventions for personal URL, the database system receives information including a user alias. The database system processes the information to determine if a user alias translation including a current network address for the user alias exists. The database system then modifies the information with the current network address using the user alias translation.

In another aspect of the inventions for predictive caching, the access server receives a request for data. The access server then determines if the data exists in a user cache wherein the user cache contains cached data based on the user's predictive patterns. The access server retrieves the data from the user cache in response to the determination that the data exists in the user cache. The access server transmits the request for data for the service provider in response to the determination that the data does not exist in the user cache.

In another aspect of the inventions for user controlled caching, the access server receives a request for data. The access server determines if the data exists in a user cache wherein the user cache contains cached data based on a user's script of commands. The access server retrieves the data from the user cache in response to the determination that the data exists in the user cache. The access server then transmits the request for data for the service provider in response to the determination that the data does not exist in the user cache.

In another aspect of the inventions for service usage audit, the access server receives an audit message into a database system. The access server processes the audit message to store the audit message in the database system.

In another aspect of the inventions for switching access by a user, switching access by a service provider, and dynamic access control, the database system receives a request. The database system processes the request to determine if the switching of the access is allowed. The database system then generates an instruction to switch access in response to the determination that the switching is allowed.

In another aspect of the inventions for network failover, network busy forwarding, time-out, busy flag, forwarding, and network endpoint availability management, the access server receives information for a destination network device. The access server determines if the destination network device is available. The access server performs an action in response to the determination that the destination network device is unavailable.

6

In another aspect of the inventions for scheduled alias translation, the access server receives information including an alias. The access server processes the information to determine whether an alias translation exists based on an alias translation schedule. The access server then modifies the information based on the alias translation in response to the determination the alias translation exists.

In another aspect of the inventions for service capability monitor, the database system receives information from an access server during a service session. The database system determines a current state of the service session based on the information. The database system determines a state transition based on the current state and a map of state transitions of the service. The database system determines whether the state transition is valid for the service session.

BRIEF DESCRIPTION OF THE DRAWINGS

The same reference number represents the same element on all drawings.

FIG. 1 illustrates a public telephone network in the prior art.

FIG. 2 illustrates a data network in the prior art.

FIG. 3 illustrates an Internet Service Provider in the prior art.

FIG. 4 illustrates conventional network access.

FIG. 5 illustrates a network architecture in an example of the invention.

FIG. 6 illustrates an access network in an example of the invention.

FIG. 7 illustrates a table for a user access profile in an example of the invention.

FIG. 8 illustrates a flowchart for an access server for inheriting a user access profile in an example of the invention.

FIG. 9 illustrates a flowchart for a database system for inheriting a user access profile in an example of the invention.

FIG. 10 illustrates a flowchart of an access server for executing a network shell in an example of the invention.

FIG. 11 illustrates a flowchart of a database system for updating a network shell in an example of the invention.

FIG. 12 illustrates a flowchart for the services based directory in an example of the invention.

FIG. 13 illustrates a flowchart of user access profile mobility in an example of the invention.

FIG. 14 illustrates a logical view of device, user, and service sessions in an example of the invention.

FIG. 15 illustrates a flowchart for a user based session in an example of the invention.

FIG. 16 illustrates a flowchart for a service based session in an example of the invention.

FIG. 17 illustrates a flowchart for a firewall/router for service capability firewall in an example of the invention.

FIG. 18 illustrates a flowchart for a database system for service capability firewall in an example of the invention.

FIG. 19 illustrates a flowchart for prepaid access in an example of the invention.

FIG. 20 illustrates a flowchart for bank card access for a connection in an example of the invention.

FIG. 21 illustrates a flowchart for network access cards for a disconnection in an example of the invention.

FIG. 22 illustrates a flowchart for network access cards for a connection in an example of the invention.

7

FIG. 23 illustrates a flow chart for network access cards for a disconnection in an example of the invention.

FIG. 24 illustrates a flowchart for global access in an example of the invention.

FIG. 25 illustrates a flowchart for an access server for user based proxies in an example of the invention.

FIG. 26 illustrates a flowchart for a user proxy for user based proxies in an example of the invention.

FIG. 27 illustrates a flowchart for an access server for subscriber based proxies in an example of the invention.

FIG. 28 illustrates a flowchart for a subscriber proxy for subscriber based proxies in an example of the invention.

FIG. 29 illustrates a flowchart for a dynamic proxy for dynamic proxies in an example of the invention.

FIG. 30 illustrates a block diagram of the access execution environment in an example of the invention.

FIG. 31 illustrates a flow chart for the access execution environment in an example of the invention.

FIG. 32 illustrates a flowchart for an access server for domain name scoping in an example of the invention.

FIG. 33 illustrates a flowchart for a database system for domain name scoping in an example of the invention.

FIG. 34 illustrates a flowchart for an access server for an inband domain name service lookup in an example of the invention.

FIG. 35 illustrates a flowchart for inline access service triggering in an example of the invention.

FIG. 36 illustrates a flowchart for an access server for access service triggering in an example of the invention.

FIG. 37 illustrates a flowchart for a database system for access service triggering in an example of the invention.

FIG. 38 illustrates a flow chart for the personal URL lookup in an example of the invention.

FIG. 39 illustrates a flow chart for the personal URL update in an example of the invention.

FIG. 40 illustrates a flowchart for an access server for auditing in an example of the invention.

FIG. 41 illustrates a flowchart for a database system for auditing in user predictive caching in an example of the invention.

FIG. 42 illustrates a flowchart for a database system for caching in user predictive caching in an example of the invention.

FIG. 43 illustrates a flowchart for a database system for auditing in user controlled caching in an example of the invention.

FIG. 44 illustrates a flowchart for a database system for caching in user controlled caching in an example of the invention.

FIG. 45 illustrates a flowchart for switching access by a user in an example of the invention.

FIG. 46 illustrates a flowchart for switching access by a service provider in an example of the invention.

FIG. 47 illustrates a flowchart for dynamic switching access in an example of the invention.

FIG. 48 illustrates a flowchart for an access server for network address failover in an example of the invention.

FIG. 49 illustrates a flowchart for a database system for network address failover in an example of the invention.

FIG. 50 illustrates a flowchart for an access server for network busy forwarding in an example of the invention.

FIG. 51 illustrates a flowchart for a database system for network busy forwarding in an example of the invention.

8

FIG. 52 illustrates a flowchart for an access server for a busy flag when the destination network device is busy in an example of the invention.

FIG. 53 illustrates a flowchart for an access server for forwarding if the destination network device timeouts in an example of the invention.

FIG. 54 illustrates a flowchart for an access server for schedule alias resolution in an example of the invention.

FIG. 55 illustrates a flowchart for a database system for scheduled alias resolution in an example of the invention.

FIG. 56 illustrates a flowchart for an access server for destination controlled forwarding in an example of the invention.

FIG. 57 illustrates a flowchart for a database system for destination controlled forwarding in an example of the invention.

FIG. 58 illustrates a flowchart for an access server for network endpoint availability management in an example of the invention.

FIG. 59 illustrates a flowchart for a database system for network endpoint availability management in an example of the invention.

FIG. 60 illustrates a flowchart for a firewall/router for service capability monitor in an example of the invention.

FIG. 61 illustrates a service capability monitor software architecture for a service capability monitor in an example of the invention.

FIG. 62 illustrates a flowchart for the network logic for service capability monitor in an example of the invention.

FIG. 63 illustrates a flowchart for the service logic for service capability monitor in an example of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Communication Network Architecture—FIGS. 5 and 6

The following description and associated figures discuss specific examples intended to teach the present invention to those skilled in the art. Those skilled in the art will appreciate numerous variations from these examples that do not depart from the scope of the invention. Those skilled in the art will also appreciate that various features described below could be combined in various ways to form multiple variations of the invention.

FIG. 5 illustrates a network architecture 500 in an example of the invention. The network architecture 500 comprises a service network 530, a service network 540, and an access network 520. The access network 520 comprises a database system 522, an access server 524, a firewall/router 526, a firewall/router 556, and an access server 554. A user network 510 includes a network device 512. The user network 510 is connected to the access server 524. The access server 524 is connected to the firewall/router 526 and the database system 522. The firewall/router 526 is connected to the service network 530, the service network 540, and the database system 522. The firewall/router 556 is connected to the service network 530, the service network 540, the database system 522, and the access server 554. The access server 554 is connected to the database system 522 and the user network 560. The user network 560 comprises a network device 562 and a network device 564.

The access network 520 provides an interface between the user network 510 and 560 and the service networks 530 and

540. The interface function provides user access profiles, security, switching, and caching. The user network 510 and 560 could be a residential or business communication system that includes network devices. A network device 512, 562, and 564 could be any device configured to exchange data or information with the access network 520. Some examples of network devices are wireless and wireline telephones, computers, modems, servers, and/or data terminals, along with associated interconnections and network interfaces. For illustrative purposes in the examples below, a user interacts with the network device 512 to access services provided by the network architecture 500 and the user network 560. The user network 510 could also be a communication destination for other users. In these cases, the access network 520 provides destination performance monitoring and control. The example of a user interacting with the network device 562 and 564 to access services provided by the network architecture 500 and the user network 510 is not discussed below for the sake of clarity.

The access server 524 and the database system 522 could be adapted from components used in current ISPs, and the access network 520 could be integrated into these ISP components. The database system 522 houses user access profiles for the users in the user networks 510 and 560 with external access rights. The user access profiles provide a data clearinghouse for user-related information for security, service options, current state, and customized macros. The service networks 530 and 540 could be voice or data systems, such as the public telephone network, Internet, public data networks, and private data networks.

When a user requests access to services, the access network 520 processes the user access profile for the user. The access network 520 performs security measures to validate the user. The access network 520 then binds the user to a terminal and to a service. This user-terminal-service binding correlates the user's capabilities in the user access profile with the capabilities of the user's current terminal and the current capabilities of the service provider. The access network 520 activates a network shell for the user and the user's terminal. The network shell is an interface that is customized for the user and the terminal. The user invokes services using the network shell.

The user access profile may contain several macros that can be as simple as address translations or as complex as lengthy computer programs. The user's network shell provides access to the user's macros. The user access profile also stores caching instructions. The caching instructions control the collection and storage of information within the access network 520 for immediate access by the user.

FIG. 6 depicts an access network in an example of the invention. The access network 520 comprises the database system 522, the access server 524, the firewall/router 526, the firewall/router 556, and the access server 554. The database system 522 comprises a local database system 570, a Lightweight Directory Access Protocol (LDAP) interface system 571, a central database system 580, a local database system 590, and an LDAP interface system 591. The local database system comprises a user profile system 572, an audit database system 573, a cache database system 574, a host system 575, a security server 576, a user authorization system 577, an alias translation system 578, and a personal DNS system 579. The central database system 580 comprises a user authorization system 581, a financial interface 582, and a cross connect system 583. The local database system 590 comprises a user profile system 592, a user authorization system 593, and an availability system 594.

The access server 524 and the firewall/router 526 are connected to the local database system 570. The local

database system 570 is connected to the central database system 580 and the LDAP interface system 571. The central database system 580 is connected to the local database system 590. The local database system 590 is connected to the firewall/router 556, the access server 554, and the LDAP interface system 591.

An access provider is an entity that provides access to users who use communications services. A typical access provider comprises an access server, a firewall/router, and a local database system. The systems within the local database system 570 could be included within the local database system 590. Also, the systems within the local database system 590 could be included within the local database system 570. As discussed above, the user is accessing the network architecture through the user network 510. The duplication of systems in the local database system 570 and the local database system 590 is excluded for the purposes of clarity. A service provider is an entity that provides communication services to users who are accessing the service through an access provider.

Network User Access Profile—FIGS. 5, 6, 7, 8, 9, 10, 11, and 12

User Access Profile Inheritance

The user access profile is stored in the access network 520 and controls user access to services. The user access profile is any information or data associated with controlling user access to a service such as role identification, authorization, billing information and access preferences.

FIG. 7 depicts table for a user access profile in an example of the invention. A user access profile includes access information for the user, billing information, and preferences for access. Access information is any information or data related to providing the user access to the network architecture 500. Some examples of access information are user ID, password, name, account number, user alias, current network address, switching allowed flag, and other security information. Access information also include a list of services that the user has subscribed to or is allowed access to. In one embodiment, access information includes a cache of information that the user has accessed previously. Some examples of billing information are address and billing code including bank card numbers or prepaid account codes. Preferences for access allow the user to save choices or preferences to customize their access to the network architecture 400. Some examples of preferences for access are file formats and Quality of Service values. The user access profile may also include usage information such as time of day access, day of week, usages per day, usages per week, and usages per month.

Users typically set up their user access profiles when signing up for the access to the network architecture 500. In one embodiment, users create the user access profile through an inheritance process. Through the inheritance process, the user selects user profile information from other user access profiles in the network architecture 500. The user may inherit user profile information from user profile data structures such as templates, other user access profiles, the user's group or class, or other networks that the user is set up in. A user profile data structure is any user profile information to be retrieved when inheriting a user access profile.

In a prior solution, a command interpreter in a single computer operating system shell environment allows a user to customize the interpretation of command strings. The user may inherit portions of other user's shell environment by adding the other user's shell attributes. User access profiles are typically stored in the access provider's database. FIGS.

5 and FIG. 8 disclose one embodiment for inheriting user access profiles in an example of the invention. The user access profile is created through an inheritance process where the user is able to select capabilities, macros, functions, methods, and data to inherit from other profiles. In this embodiment, users inherit user access profiles from classes, groups, or provider recommendations. Features of user access profiles such as alias translation could then be implemented with new users rapidly. The inheritance of user access profiles also simplifies the configuration of users. In one example, a user initiates the user access profile inheritance by clicking a button on a website. Also, when a user requests a new service, the service provider automatically inherits the user access profile for the user so the user is able to use the requested service.

FIG. 8 depicts a flowchart for the access server 524 for inheriting a user access profile in an example of the invention. FIG. 8 begins in step 800. In step 802, the access server 524 waits for the next packet. The access server 524 then receives and processes the packet from the network device 562 in step 804. The access server 524 then checks if the destination is the user access profile in step 806. If the destination is not the user access profile, the access server 524 transmits the packet on the correct path in step 808 before returning to step 802. If the destination is the user access profile, the access server 524 then checks if the network device 562 is allowed access to the user access profile in step 1110. If the network device 512 is not allowed access to the user access profile, the access server 524 discards the packet and registers a network request security event in step 812 before returning to step 802.

If the network device 512 is allowed access to the user access profile, the access server 524 then checks if the user is allowed to update their user access profile in step 814. If the user is not allowed updates to their user access profile, the access server 524 generates and transmits a profile update not allowed message to the network device 512 in step 816 before returning to step 802. If the user is allowed updates to their user access profile, the access server 524 generates and transmits a user access profile update permission message asking if the user wishes to update the user access profile to the network device 512 in step 818. The access server 524 then checks if the user approved the user access profile update in step 820. If the user does not approve, the access server 524 generates and transmits a user access profile aborted message to the network device 512 in step 822 before returning to step 802.

If the user approves the user access profile update, the access server 524 sets an external request timer in step 824. The access server 524 then generates and transmits a user access profile update request to the database system 522 in step 826. A user access profile update request could be any signaling, message, or indication to update the user access profile. The access server 524 then checks if a reply was received or the external request timer expired in step 828. If the reply was not received and the external request timer did not expire, the access server 524 returns to step 828. If the reply was received or the external request timer did expire, the access server 524 checks if the reply was valid in step 830. If the reply was valid, the access server 524 generates an update complete message in step 832 before returning to step 802. If the reply was not valid, the access server 524 discards the packet and replies to the network device 512 that the user access profile update failed in step 834 before returning to step 802.

FIG. 9 depicts a flow chart for the database system 522 for inheriting a user access profile in an example of the inven-

tion. FIG. 9 begins in step 900. In step 902, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 904. In step 906, the database system 522 then checks if the access server requester is known. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 912. The database system 522 appends the reply to the packet and transmits the reply to the access server 524 in step 910. The database system 522 then replies with a decline message to the access server 524 in step 912 before returning to step 902.

If the requester is known, the database system 522 then checks whether the request is a user access profile update request in step 914. If the request is not a user access profile update request, the database system 522 checks if the request is a security event in step 916. If the request is not a security event, the database system 522 then registers an unknown action event in step 918 before returning to step 910. If the request is a security event, the database system 522 increments a path/device security record in step 924. The database system 522 then logs the path/device security record in step 926 before returning to step 902.

If the request is a user access profile update request, the database system 522 identifies and authenticates the user and the network device 512 in step 920. The database system 522 then checks if the access is valid in step 922. If the access is invalid, the database system 522 returns to step 924. If the access is valid, the database system 522 retrieves the user access profile information from a user profile data structure in step 928. The retrieval of user access profile information may be based on any information in the user access profile such as group or class or selections of inheritance user access profile presented to the user. The group or classes could be any logical grouping of user based on similar interests or situations such as work, family, and geographic location. The database system 522 then updates the user access profile in step 930 based on the user access profile information retrieved in step 928 and the user's selections for updating. The database system 522 then replies with an approve message to the access server 524 in step 932 before returning to step 902. In one embodiment of the invention, the database system 522 uses the user access profile system 572 to perform the operations disclosed in FIG. 9 and stores the user access profile in the user access profile system 572.

45 Network Shell

The user access profile includes data to provide a customized network shell to the user. The network shell is a user interface to the access network that is linked to programs, methods, macros, or service. Typically, configuration of the network shell is graphically presented to the user on a display. For example, the network shell appears as a list of alias selections that are associated with actions such as a program or macro for resources or services. An alias selection is any information that is associated with an action to be executed when the alias selection is selected. In another example, the alias selections are graphically presented as icons that relate to actions to be executed. The network shell overlays the standard DNS offered in IP networks, which reduces alias translation delays and required user keystrokes. In prior solutions, a command interpreter in an operating system shell environment allows a user to customize the interpretation of command strings, or a DNS translates "alias" values to addresses. FIGS. 5, 10, and 11 show one embodiment for network shells in an example of the invention.

FIG. 10 depicts a flowchart of an access server for executing a network shell in an example of the invention.

13

FIG. 10 begins in step 1000. In step 1002, the access server 524 waits for the next packet. The user selects an alias selection from the network shell graphically presented. In another embodiment, the user enters an alias value by a non-graphical means. Alternatively, the network shell is not presented to the user. The network device 512 transmits a packet including the alias selection to the access server 524. In step 1004, the access server 524 receives and processes the packet from the network device 512 and processes the packet. The access server 524 then checks if the network device 512 is allowed access to the network architecture 500. If the network device 512 is not allowed access to the network architecture 500, the access server 524 discards the packet and registers a network security event in step 1008 before returning to step 1002.

If the network device 512 is allowed access to the network architecture 500, the access server 524 checks if the user is recognized in step 1010. If the user is not recognized, the access server 524 returns to step 1008. If the user is recognized, the access server 524 retrieves the user's network shell in step 1012. In some embodiments, the user's network shell is retrieved from the user access profile in the access database 522 prior to presenting the network shell to the user. In step 1014, the access server 524 checks if the packet from the network device 512 includes an alias selection from the user's network shell. In one embodiment, specialized hardware is used to scan for aliases just as IP addresses are currently scanned for. If the packet does not include an alias selection from the user's network shell, the access server 524 proceeds to step 1018. If the packet does include the alias selection from the user's network shell, the access server 524 executes the action associated with the alias selection in step 1016 before returning to step 1002. In step 1018, the access server 524 processes the packet with the normal handling before returning to step 1002.

FIG. 11 depicts a flowchart of a database system for updating a network shell in an example of the invention. FIG. 11 begins in step 1120. In step 1122, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 1124. The database system 522 then checks if the access server requester is known in step 1126. If the access server requester is not known, the database system 522 registers an unknown requester event in step 1128. The database system 522 then appends the reply to the packet and transmits the packet to the access server 524 in step 1130. In step 1132, the database system 522 replies with a decline message to the access server 524 before returning to step 1122.

If the access server requester is known, the database system 522 checks if the request is a profile update for the network shell in step 1134. If the request is a profile update for the network shell, the database system 522 identifies and authenticates the user and the network device 512 in step 1136. The database system 522 then checks whether the access is valid in step 1138. If the access is invalid, the database system proceeds to step 1154. If the access is valid, the database system 522 retrieves the user access profile in step 1140. The database system 522 then updates the user access profile with the network shell with the user's alias selections and the associated programs, macros, functions or methods in step 1142. The database system 522 then replies with an approve message in step 1144 before returning to step 1122.

If the request is not a profile update, the database system 522 checks if the request is an action event in step 1146. If the request is an action event, the database system 522

14

performs the action and replies in step 1148 before returning to step 1122. If the request is not an action event, the database system 522 checks if the request is a security event in step 1150. If the request is not a security event, the database system 522 registers an unknown request type event in step 1152 before returning to step 1122. If the request is a security event, the database system 522 proceeds to step 1154. In step 1154, the database system increments a path/device security record. The database system 522 then appends the requester information to the packet and logs the event before returning to step 1122. In one embodiment of the invention, the database system 522 uses the user access profile system 572 to perform the operations disclosed in FIG. 11 and stores the user access profile in the user access profile system 572.

Service Based Directory

In prior systems, the RADIUS server provided the user with selections for transport modes. However, the user was not able to select available network based services. FIGS. 5 and 12 disclose one embodiment for a service based directory in an example of the invention. In this embodiment, access providers provide a list of services that the user can select. With the list of services, the access providers have the ability to advertise specific services to users. The list of services may be generated based on the user access profile to make the list user specific. Once the user makes the selection, the access server 524 connects the user to the service network such as Intranet, Internet, or private dedicated network that provides the selected service.

FIG. 12 depicts a flowchart for the services based directory in an example of the invention. FIG. 12 begins in step 1200. In step 1202, the access server 524 waits for the next connection from a network device in the user network 510. The access server 524 then receives a connection from the network device 512 in step 1204. Once the connection is established, the access server 524 generates and transmits a user ID query to the network device 512 in step 1206. The access server 524 then receives an ID reply and establishes a network device session in step 1208.

The access server 524 then generates an available services reply including a list of services in 1210. In one embodiment, the access server 524 generates the available services reply based upon information in the user access profile. The access server 524 receives a selected service reply from the network device 512 in step 1212. The access server 524 then connects the network device 512 to the selected service provider in step 1214. The access server 524 waits for the next packet in step 1214. The access server 524 then exchanges packets between the network device 524 and the selected service provider in step 1218.

Access Network User Binding—FIGS. 5, 13, 14, 15, and 16

User Access Profile Mobility

Users may access their user access profile from any network device connected to the network architecture 500. In a prior solution, a command interpreter in an operating system shell environment allows a user to customize the interpretation of command strings. In this environment, the access network identifies the user and the terminal device interface, which provides user mobility. The access network then executes customized actions for the user. FIGS. 5 and 13 show one embodiment of the invention for user access profile mobility. This embodiment provides user mobility in a distributed data network environment.

FIG. 13 depicts a flowchart of user access profile mobility in an example of the invention. FIG. 13 begins at step 1300. A user at the network device 512 signs on the access network

520 with their user ID. The network device 512 transmits user information with the user ID to the access server 524. In this embodiment, the user information is in the form of a packet. In step 1302, the access server 524 receives and processes the packet to check if the user access profile is local within the local database system 570. A user access profile is local within the local database system 570 when the user access profile is located in the local database system 570. If the user access profile is local, the access server 524 proceeds to step 1312.

If the user access profile is not local within the local database system 570, the access server 524 checks if the user ID is delimited with a provider ID in step 1304. One example of a user ID delimited with a provider ID includes a user's name and a provider ID separated by a delimiter such as joesmith@access.net. If the user ID is not delimited, the access server 524 retrieves the location of the default user access profile system using a default Lightweight Directory Access Protocol (LDAP) interface system 571 in step 1308 before proceeding to step 1312.

If the user ID is delimited, the access server 524 checks if the provider ID is valid in step 1306. If the provider ID is not valid, the access server 524 registers an invalid profile request security event before terminating at step 1326. If the provider ID is valid, the access server 524 uses the provider ID to retrieve the location of the local database system 590 from a foreign LDAP interface system 591 before proceeding to step 1312.

In step 1312, the access server 524 checks if the packet is a retrieve request for the user access profile. If the packet is a retrieve request, the access server 524 generates and transmits a request to retrieve the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then creates the user session ID and updates the user entries of the user access profile with the user ID and date/time updated in step 1312. The access server 524 then transmits a reply message with the session ID and the location of the LDAP server in step 1314 before terminating at step 1326.

If the packet is not a retrieve request, the access server 524 checks if the packet is a release request in step 1316. If the packet is a release request, the access server 524 retrieves the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then updates the user session ID and updates the user entries of the user access profile with the user ID and date/time updated in step 1318. The access server 524 then transmits a reply message with the session ID in step 1320 before terminating at step 1326.

If the packet is not a release request, the access server 524 checks if the packet is an update request in step 1322. If the packet is not an update request, the access server 524 registers an invalid profile request security event before terminating at step 1326. If the packet is an update request, the access server 524 retrieves the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then updates the user access profile with the information in the packet in step 1324. The access server 524 then transmits a complete message to the user network 510 to signify the profile update is complete before terminating at step 1326. In one embodiment, the local database system 570 uses the user access profile system 572 to retrieve and store the local user access profiles and the local database system 590 uses the user access

profile system 592 to retrieve and store the foreign user access profiles.

User, Device, and Service Sessions

Access network providers provide user and device sessions in addition to service sessions to distinguish users when providing communication services. A user session is the information associated with a user accessing a network. A device session is the information associated with a device being used to access a network. A service session is the information associated with a service being provided over a network. Service providers distinguish users instead of access devices or links. This allows multiple users to share a single access device with each user receiving their own customized or preferred services. Advantageously, service providers establish service access rights and restrictions such as preventing adult content for younger viewers or sharing an access device for business and personal use. Also, user, device, and service sessions allow a service provider to group multiple service providers to provide a composite of services to the user similar to a contractor/sub-contractor relationship.

FIGS. 5, 14, 15, and 16 disclose one embodiment for device, user, and service sessions in an example of the invention. FIG. 14 depicts a logical view of device, user, and service sessions in an example of the invention. Devices 1402, 1404, and 1406 are comprised of session type specific information and session links. Session type specific information include public keys, private keys, and session ID. Session links include user sessions, device sessions, and service sessions. The public keys and private keys are for encryption and decryption of messages. Session ID identifies the session ID for the device. User sessions are user sessions that the device is logically linked to. Service sessions are the service sessions the device is logically linked to. Users 1408, 1410, and 1412 are comprised of public keys, private keys, session ID, device sessions, and service sessions. Session ID identifies the session ID for the user. Device sessions are the device sessions the user is logically linked to. Service sessions are the service sessions the user is logically linked to. Services 1414, 1416, and 1418 are comprised of public keys, private keys, session ID, user sessions, device sessions, and sub service sessions. Session ID identifies the session ID for the service. User sessions are the user sessions the service is logically linked to. Device sessions are the device sessions the service is logically linked to. Sub service sessions are the service sessions the service is logically linked to.

In one example, device 1402 is linked to user B 1410 via a link 1422. User B is also linked to service N 1418 via a link 1424. Device 1402 contains the user information in the user session fields for user B 1410 and the service information in the service session fields for service N. User B 1410 contains the device information in the device session fields for device 1402 and the service information in the service session fields for service N 1418. Service N 1418 contains the device information in the device session fields for device 1402 and the user information in the user session fields for user B 1410. Service N 1418 is also linked to service A 1414 via a link 1426. Service N 1418 contains additional service information in the service session fields for service A 1414. Service A 1414 contains the composite service information in the service session fields for service N 1418. Service A 1414 also includes an owning service which is a reference to the service that owns service A.

FIG. 15 depicts a flowchart for a user based session in an example of the invention. FIG. 15 begins in step 1500. In step 1502, the network device 512 establishes a connection

17

with the access server 524. The access server 524 generates and transmits a user ID query to the user network 510 in step 1504. In step 1506, the network device 512 receives the user ID query and transmits a user ID reply to the access server 524. The access server 524 receives and processes the user ID reply to generate a device session. The network device 512 then transmits a packet to the access server 524. The access server 524 then receives the packet in step 1508. In step 1510, the access server 524 processes the packet to check if the packet includes a user session ID.

If the packet does not include the user session ID, the access server 524 transmits a login query encrypted by the network device's 512 public encryption key 1402 to the network device 512 in step 1512. The network device 512 decodes the login query with its public encryption key and transmits a login reply encrypted with its public encryption key in step 1516. The access server 524 then decodes the login reply with the private encryption key and checks if the user ID is valid. If the user is valid, the access server 524 generates a user session and a session ID in step 1520. In some embodiments, the user session ID is the original IP address. The access server 524 then encrypts with the public encryption key and transmits a complete reply with the user session ID to the network device 512 in step 1522 before returning to step 1508.

If the packet does include the user session ID, the access server 524 checks if the user and user session ID are valid in step 1514. If the user or user session ID is not valid, the access server 524 discards the packet and registers a security event in step 1515 before returning to step 1508. If the user and user session ID are valid, the access server 524 generates a service request with the user session ID and the service session ID if available. The access server 524 encrypts the service request with the public encryption key where appropriate. The access server 524 transmits the packet including the service request in step 1518 before returning to step 1508.

FIG. 16 depicts a flowchart for a service based session in an example of the invention. FIG. 16 begins in step 1600. The network device 512 transmits a service request to the access server 524. The access server 524 then receives the service request in step 1602. In step 1604, the access server 524 checks if the service request includes a service session ID.

If the service request does not include the service session ID, the access server 524 transmits a service ID query encrypted with the public encryption key to the network device 512 in step 1606. The network device 512 decodes the service ID query with its private key and transmits a service reply encrypted with the service public encryption key to the access server 524 in step 1610. The access server 524 then decodes the service reply with the service private encryption key and checks if the user is valid. If the user is valid, the access server 524 generates a service session and a session ID in step 1614. The access server 524 then transmits a complete reply with the service session ID to the network device 512 in step 1616 before returning to step 1602.

If the packet does include the service session ID, the access server 524 checks if the user and service session ID are valid in step 1608. In one embodiment, the service session ID is the destination IP address. If the user or service session ID is not valid, the access server 524 discards the packet and registers a security event before returning to step 1602. If the user and service session ID is valid, the access server 524 updates the service request with the user session ID and the service session ID. The access server 524

18

encrypts the service request with the service public encryption key where appropriate. The access server 524 transmits the service request with the user session ID and the service session ID in step 1612 before returning to step 1602.

Access Network Security—FIGS. 5, 17, and 18 Service Capability Firewall

A network service provider interfaces with a network access provider at a transport functional level. The interface typically includes Internet protocol firewalls to provide security. Unfortunately, the interface between the network service provider and the network access provider is not able to hide the implementation details such as addressing schemes, transport details, and equipment specifics. The hiding of implementation details is preferred to prevent hackers from manipulating the implementation details. FIGS. 5, 17 and 18 depict one embodiment for a service capability firewall in an example of the invention. In this embodiment, the interface between the network service provider and the network access provider occurs at a named functional level instead of the transport addressing level. A named function request is any request for a capability of service provided by the network service provider. This allows the network service provider to hide the implementation details. The network service provider exposes only functional capabilities to the users depending on their security rights.

FIG. 17 depicts a flowchart for the firewall 556 for service capability firewall in an example of the invention. FIG. 17 begins in step 1700. In step 1702, the firewall 556 waits for information. In this embodiment, the information is in the form of a packet. The firewall 556 receives and processes a packet including a named function request from the network device 512 in step 1704. The firewall 556 then checks whether the firewall 556 is the destination for the named function request in step 1706. If the firewall 556 is not the destination, the firewall 556 registers a network request security event in step 1710. In step 1712, the firewall 556 encapsulates the packet with path information and transmits the packet to the database system 522 before returning to step 1702.

If the firewall 556 is the destination, the firewall 556 checks whether the sending address is consistent with the path in step 1714. If the sending address is not consistent with the path, the firewall 556 returns to step 1710. If the sending address is consistent with the path and does not belong to the accessing network, the firewall 556 checks if the sending address/session ID/named function combination is cached in step 1716. If the sending address/session ID/named function combination is cached, the firewall 556 replaces the packet's named function request with the private cached destination address in step 1718. The firewall 556 then checks if the private destination address is known and allowed to pass in step 1720. If the destination is known and allowed to pass, the firewall 556 transmits the packet on the correct path with standard firewall access in step 1722 before returning to step 3802. If the destination is not known or not allowed to pass, the firewall 556 returns to step 1710.

If the sending address/session ID/named function combination is not cached, the firewall 556 set an external request timer in step 1724. The firewall 556 then generates and transmits a request for the database system 522 in step 1726. The firewall 556 checks whether a reply is received or the timer has expired in step 1728. If the reply is not received and the timer has not expired, the firewall 556 returns to step 1728. In another embodiment, a blocked I/O is used instead of the wait loop in step 1728. If the reply is received or the timer has expired, the firewall 556 checks if there is a valid

reply in step 1730. If the reply is invalid or no reply was received, the firewall 556 discards the packet and registers a translation failure event in step 1734 before returning to step 1702. If the reply is valid, the firewall 556 replaces the packet's named function request based on the reply and caches the address/session functions returned in step 1732 before returning to step 1720.

FIG. 18 depicts a flowchart for the database system 522 for service capability firewall in an example of the invention. FIG. 18 begins in step 1800. In step 1802, the database system 522 waits for the next request. The database system 522 then receives and processes the request in step 1804. In step 1806, the database system 522 then checks if the access server requester is known. If the access server requester is not known, the database system 522 registers an unknown requester event in step 1808. The database system 522 then appends the reply with the packet and replies to the access server 524 in step 1810. The database system 522 then generates and transmits a decline reply to the access server 524 in step 1812 before returning to step 1802.

If the requester is known, the database system 522 then checks if the request is a capability appeal in step 1814. A capability appeal is any message, signaling or instruction requesting a capability with a related network address in the service provider. If the request is not a capability appeal, the database system 522 then checks if the request is a security event in step 1816. If the request is not a security event, the database system 522 register an unknown request event in step 1817 before returning to step 1810. If the request is a security event, the database system 522 increments a path/device security record and appends the requester information to the request packet and logs the event in step 1822 before returning to step 1802. If the request is a capability appeal, the database system 522 then identifies the user and network device 512 in step 1818. The database system 522 then checks if the access is valid in step 1822. If the access is invalid, the database system 522 returns to step 1822. If the access is valid, the database system 522 retrieves the user and network device profiles in step 1824. The database system 522 then checks whether the capability is valid for the user and session state in step 1826. If the capability is invalid, the database system 522 returns to step 1822. If the capability is valid, the database system 522 generates a session ID, the network address of the capability requested, and functions available and appends the network address, which is only sent to the firewall/router 526 and not to the user, to the reply in step 1830. The database system 522 then transmits the approve reply to the access server 524 in step 1832 before returning to step 1802.

Access Network Service Authorization—

FIGS. 5, 19, 20, 21, 22, 23, and 24

Prepaid Access

Prepaid phone cards are commonly used in PSTN, where the customer pays a prepaid amount that is debited against when the customer makes a call. In data networks, prepaid cards are not currently being used. FIGS. 5 and 19 disclose one embodiment for prepaid access in an example of the invention. In this embodiment, users buy prepaid cards from network providers. When the user requests access to one of many access providers throughout the country, the access provider verifies the prepaid account code before providing the access. A prepaid account code is any number that relates to a user's prepaid account. The prepaid account is debited against for the charges related to the access. Other charges related to the service provided may also be debited against the prepaid account. For example, a user may purchase an item from a website and have the charges debited against the

prepaid account. Once the prepaid amount is reached, the access provider terminates the access to the user. In one embodiment, the network provider provides different levels of service such as gold, silver, and bronze. The gold service has guaranteed throughput but higher rates for access, while the bronze service has lower throughput and rates.

FIG. 19 depicts a flowchart for prepaid access in an example of the invention. FIG. 19 begins in step 1900. In step 1902, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 1904. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits information identifying a billing code. A billing code is any number that identifies a user for billing. Two examples of billing codes are prepaid account codes and credit card numbers. In this embodiment, the information identifying a billing code is a response including a prepaid account code to the login query. In other embodiments, the billing code is retrieved from the user access profile. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response including the prepaid account code to determine if the user is known in the local database system 520 in step 1906. A user is known when the user is allowed to use the access network 520. In one embodiment, the user is known in the local database system 520 if there is time/amounts left in the user's prepaid account. In another embodiment, the database system 522 evaluates a positive balance file to determine the remaining time/amount in the user's prepaid account. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 1908 before returning to step 1902.

If the user is not known, the database system 522 checks if there is an appeal server for user authentication in step 1910. In one embodiment, the access server 524 performs the appeal on a decline. If there is no appeal server, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2412 before returning to step 2402. If there is an appeal server, the database system 522 generates an authorization query including the prepaid account code for the central database system 580 in step 1914. The database system 522 checks if the user is known in the central database system 580 in step 1914. In one embodiment, the user is known in the central database system 580 if there is time/amounts left in the user's prepaid account. If the user is known, the database system 522 proceeds to step 1908. If the user is not known, the database system 522 proceeds to step 1910.

In one embodiment, the database system 522 uses a user authorization system 575 for checking if the user is known in the local database system 520. The user authorization system 575 contains all the prepaid customers, prepaid customer information, prepaid account codes, and the amount/quantity remaining in the prepaid account to verify if access is allowed. In one embodiment, the database system 522 uses a user authorization system 581 as the appeal server for checking if the user is known in the local database system 580. The user authorization system 581 contains all the prepaid customers, prepaid customer information, and the amount/quantity remaining in the prepaid account.

Bank Card Access

In prior systems, access providers authenticated users using their own databases. FIGS. 5, 20 and 21 disclose one

embodiment for bank card access in an example of the invention. In this embodiment, access providers authenticate users through bank card financial networks using the users' credit card numbers. Network providers use credit or debit card numbers as user ID and passwords for authentication and authorization purposes. Users use prepaid cards, phone cards, and credit cards in PSTN. However, no bank cards have been used for access to data networks other than for batch bill payment.

FIG. 20 depicts a flowchart for bank card access for a connection in an example of the invention. FIG. 20 begins in step 2000. In step 2002, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2004. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits information identifying a billing code. In this embodiment, the information identifying a billing code is a response including credit card numbers to the login query. In other embodiments, the billing code is retrieved from the user access profile. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response to see if the user is known in the local database system 570 in step 2006. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 2008 before returning to step 2002.

If the user is not known, the database system 522 processes the response to check if the user is identified by credit card numbers in step 2010. If the user is not identified by the credit card numbers, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2012 before returning to step 2002. If the user is identified by the credit card numbers, the database system 522 generates an authorization query as a pre-authorization hold or authorization/capture transaction for the central database system 580 in step 2014. In one embodiment, the central database system 580 uses a financial interface 582 to interface with a financial switch to banks for authentication and authorization. The database system 522 then checks if the user is authenticated and authorized by the central database system 580 in step 2016. If the user is authenticated and authorized, the database system 522 logs access information and an authorizing financial entity or institution in step 2018 before proceeding to step 2008. If the user is not known, the database system 522 checks if there is another database system for authorization such as for foreign user access in step 2018. If there is another database system, the database system 522 returns to step 2014. If there is not another database system, the database system 522 returns to step 2012.

FIG. 21 depicts a flowchart for bank card access for a disconnection in an example of the invention. FIG. 21 begins in step 2100. In step 2102, the access server 524 waits for a disconnection. The network device 512 then disconnects from the access server 524 in step 2104. The access server 524 then checks whether the user was on a pre-authorization hold in step 2106. If the user is not on a pre-authorization hold, the 522 logs access information and an authorizing database system 522 in step 2108 before proceeding to step 2102. If the user is on a pre-authorization hold, the database system 522 generates a pre-authorization complete transaction for the central database system 580 in step 2110 before returning to step 2108.

Access Cards

In prior systems, access providers authenticated users using their own databases. FIG. 5, 22 and 23 disclose one embodiment for network access cards in an example of the invention. An access card is a card that a user uses to access a network. The access card includes an access card account code. The access card account code is any number that relates to the user's access account. In this embodiment, access providers authenticate users who have access cards using other access providers' databases. Access providers then may enter into sharing agreements with other access providers. These sharing agreements allow users to use multiple access providers, which provide the users greater mobility and availability. Users use phone cards in PSTN for phone card access. However, no access cards for data networks have been used.

FIG. 22 depicts a flowchart for network access cards for a connection in an example of the invention. FIG. 22 begins in step 2200. In step 2202, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2204. In another embodiment of the invention, the user calls a toll free number to request access. A service control point is queried to determine where to route the request for access similar to an automatic call distribution (ACD). The request for access is then routed to the access server 524 to establish a connection between the network device 512 and the access server 524.

Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits a response including the access card account code to the login query. The access server 524 then transfers the response from the network device 512 to the database system 522. In this embodiment, the database system 522 checks if the user is allowed to use the access network 520 by checking whether the user is known. A user is known when the user is allowed to use the access network 520. For example, a user is known when their user access profile is located within the local database system 570 and their user access profile allows the user to use the access network 520. The database system 522 receives and processes the response including the access card account code to check if the user is known in the local database system 570 in step 2206. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 2208 before returning to step 2202.

If the user is not known, the database system 522 checks if the response included foreign network account information in step 2210. Foreign network account information is any information that is indicative of an account that is external to local database system 570 that the user is attempting to gain access. If there is no foreign network account information, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2212 before returning to step 2202.

If there is foreign network account information, the database system 522 identifies the local database system 590 based on the foreign network account information and generates an authorization query for the local database system 590 in step 2214. The database system 522 then checks if the user is authenticated and authorized by the local database system 590 in step 2216. If the user is authenticated and authorized, the database system 522 logs contract and settlements information returned by the local database system 590 or indicated by the database system 522 in relation to local database system 590 in step 2218 before proceeding to step 2208. If the user is not known, the

23

database system 522 proceeds to step 2212. In one embodiment, the local database system 570 uses the user authorization system 575 to check if the user is known in the local database system 570. In one embodiment, the local database system 590 uses the user authentication system 593 for authentication and authorization.

FIG. 23 depicts a flowchart for network access cards for a disconnection in an example of the invention. FIG. 23 begins in step 2300. In step 2302, the access server 524 waits for a disconnection. The network device 512 then disconnects from the access server 524 in step 2304. The access server 524 then generates and transmits a logoff query for the database system 522. The database system 522 then transfers the logoff query to the local database system 570 and the local database system 590. The database system 522 logs the access information and the authorizing database system in step 2308 before returning to step 2302.

Global Authentication

In prior systems, access providers authenticated users using their own databases. FIGS. 5 and 23 disclose one embodiment for global authentication in an example of the invention. In this embodiment, access providers authenticate users using a centralized database. Access providers then may enter into sharing agreements with other access providers. These sharing agreements allow users to use multiple access providers, which provide the users greater mobility. Currently, cellular telephone companies enter into these sharing arrangements for greater coverage. However, this sharing of databases for authentication and authorization has not occurred for data networks.

FIG. 24 depicts a flowchart for global access in an example of the invention. FIG. 24 begins in step 2400. In step 2402, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2404. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits a response to the login query. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response to see if the user is native in the local database system 570 in step 2406. In this embodiment, the database system 522 checks if the user is allowed to use the access network 520 by checking whether the user is native. A user is native when the user is allowed to use the access network 520. For example, a user is native when their user access profile is located within the local database system 570 and their user access profile allows the user to use the access network 520. If the user is native, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 4308 before returning to step 2402.

If the user is not native, the database system 522 checks if there is an authentication/authorization server in the database system 522 for a foreign network for user authentication in step 1910. If there is no authentication/authorization server in the database system 522 for the foreign network, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2412 before returning to step 2402. If there is an authentication/authorization server in the database system 522 for the foreign network, the database system 522 generates an authorization query for the central database system 580 in step 2414. The database system 522 then checks if the user is known in the central database system 580 in step 2414. If the user is known, the database

24

system 522 proceeds to step 2408. If the user is not known, the database system 522 proceeds to step 2410. In one embodiment, the central database system 580 uses a user authorization system 581 to check if the user is known.

Access Network Proxy/Environment—
FIGS. 5, 25, 26, 27, 28, 29, 30, and 31
User Based Proxy

A proxy is an application that represents itself as one or more network endpoints. The proxy receives a request for services from a user at a network endpoint and acts on behalf of the user in transmitting and receiving user requests and replies. There are Internet proxy agents that are user network access bind points. However, the Internet proxy agents are not user specific, and they act to protect user interests, not network. These proxy agents provide address translation and basic firewall functionality. Client focused proxies have extended to cookie collection and password handling.

FIGS. 5, 25 and 26 disclose one embodiment for user based proxies in an example of the invention. The user based proxies obtain information for the user and establishes a user specific network presence. A benefit of having a user specific network presence is that user access is handled by a process owned by a network security certificate authority that prevents Trojan horse network attacks. User based proxy provides a single control and monitor point for a user. The proxy agents provides a bind point for all user specific access such as user profile functionality, translations, security, and caching.

FIG. 25 depicts a flowchart for the access server 524 for user based proxies in an example of the invention. FIG. 25 begins in step 2500. In step 2502, the access server 524 waits for the next event. If the next event is a connection, the network device 512 connects with the access server 524 in step 2504. In step 2506, the access server 524 then collects user information and transfers the user information to the database system 522 to perform the security. In one embodiment, the database system 522 uses a security server 576 to perform the security. The access server 524 then checks if the user is allowed access in step 2508. If the user is not allowed access, the access server 524 disconnects the user in step 2510 before returning to step 2502.

If the user is allowed access, the access server 524 checks if a proxy is available in the database system 522 in step 2512. In one embodiment of the invention, the user proxies are in the host system 575. If the proxy is available, the access server 524 proceeds to step 2516. If the proxy is not available, the access server 524 starts the proxy in the database system 522 in step 2514 before proceeding to step 2516.

The access server 524 checks if the user access profile information is available in step 2516. If the user access profile information is not available, the access server 524 generates and transmits a user profile error to the network device 512 in step 2518 before returning to step 2502. If the user profile information is available, the access server 524 configures the proxy for the user in step 2520. The access server 524 then generates and transmits a message with the address of the user proxy and the public encryption key to the network device 512 in step 2522. The access server 524 also collects and transmits the access address of the network device 512 and the public encryption key to the user proxy. The access server 524 then returns to step 2502.

If the next event is a disconnection, the network device 512 disconnects with the access server in step 2524. The access server 524 then generates and transmits a reset command to the user proxy to clear the proxy state information and configuration in step 2526 before returning to

25

step 2502. If the next event is a status/request event from the user proxy, the access server 524 sets a user proxy reply timer in step 2528. The user proxy has written a status to the access server 524, and the user proxy receives a reply in step 2530 before returning to step 2502. If the next event is a timer expiration, the access server 524 receives the user proxy reply timer expiration in step 2532. The access server 524 then generates and transmits a continue wait message and status to the user proxy in step 2534 before returning to step 2528.

FIG. 26 depicts a flowchart for a user proxy for user based proxies in an example of the invention. FIG. 26 begins in step 2600. In step 2602, the user proxy waits for the next event. If the next event is the completion of the initialization, the user proxy checks if the initialization is complete in step 2604. The user proxy then sets a request timer in step 2606. The user proxy then request work from the access server 524 and transmits a status to the access server 524 in step 2608 before returning to step 2602. If the next event is the expiration of the request timer, the user proxy checks if the request timer expired in step 2610. The user proxy then registers a timeout in step 2612 before returning to step 2606.

If the next event is an access server 524 reply, the user proxy receives the access server 524 reply in step 2614. The user proxy then checks if the reply is a continue in step 2616. If the reply is a continue, the user proxy returns to step 2606. If the reply is not a continue, the user proxy processes the configuration information or action in step 2618 before returning to step 2606. If the next event is user request, the user proxy receives the user request for service in step 2620. The user proxy then checks if the user is valid in step 2622. If the user is valid, the user proxy then checks if the request is valid in step 2624. If the request is valid, the user proxy packages the request with the security certification and encryption in step 2626. The user proxy then transmits the request to the appropriate network destination in step 2628 before returning to step 2602. If the user or request is not valid, the user proxy registers a security event in step 2630 before returning to step 2602. The user proxy exchanges user information between the user network 510 and the appropriate network destination.

Subscriber Based Proxy

Another type of proxy is a subscriber based proxy. A subscriber is a logical entity such as an organization, corporation, or other grouping of users that has subscribed to services with a service provider. FIGS. 5, 27 and 28 disclose one embodiment for subscriber based proxies in an example of the invention. The subscriber based proxies obtain information for a user of a subscriber group and establishes a subscriber specific network presence. The benefit of having a subscriber specific network presence is that user access rights of the subscriber can be handled as a group, and group rights are owned by a network security certificate authority. The subscriber proxy agents provide a bind point for all subscriber specific access such as user profile functionality, translations, security, and caching.

FIG. 27 depicts a flowchart for the access server 524 for subscriber based proxies in an example of the invention. FIG. 27 begins in step 2700. In step 2702, the access server 524 waits for the next event. If the next event is a connection, the network device 512 connects with the access server 524 in step 2704. In step 2706, the access server 524 then collects user information and transfers the user information to the database system 522 to perform the security. In one embodiment, the database system 522 uses a security server 574 to perform the security. The access server 524

26

then checks if the user is allowed access in step 2708. If the user is not allowed access, the access server 524 disconnects the user in step 2710 before returning to step 2702. If the user is allowed access, the access server 524 checks if all required subscriber proxies are available in the database system 522 in step 2712. In one embodiment of the invention, the subscriber proxies are in the host system 575. If the proxies are available, the access server 524 proceeds to step 2716. If the proxies are not available, the access server 524 starts the required proxies in the database system 522 in step 2714 before proceeding to step 2716.

In step 2716, the access server 524 checks if the subscriber access profile information is available. If the subscriber access profile information is not available, the access server 524 generates and transmits a subscriber profile error to the network device 512 in step 2718 before returning to step 2702. If the subscriber profile information is available, the access server 524 configures the subscriber proxies for the subscriber in step 2720. The access server 524 then generates and transmits a message with the address of the subscriber proxy and the public encryption key to the network device 512 in step 2722. The access server 524 also collects and transmits the access address of the network device 512 and the public encryption key to the subscriber proxies. The access server 524 then returns to step 2702.

If the next event is a disconnection, the network device 512 disconnects with the access server in step 2724. The access server 524 then generates and transmits a reset command to the subscriber proxy in step 2726 before returning to step 2702. If the next event is a status/request event from the subscriber proxy, the access server 524 sets a subscriber proxy reply timer in step 2728. In step 2730, the subscriber proxy has written a status to the access server 524, and the subscriber proxy receives a reply before returning to step 2702. If the next event is a subscriber proxy timer expiration, the access server 524 receives the subscriber proxy reply timer expiration in step 2732. The access server 524 then generates and transmits a continue wait message and a status to the subscriber proxy in step 2734 before returning to step 2730.

FIG. 28 depicts a flowchart for a subscriber proxy for subscriber based proxies in an example of the invention. FIG. 28 begins in step 2800. In step 2802, the subscriber proxy waits for the next event. If the next event is the completion of the initialization, the subscriber proxy checks if the initialization is complete in step 2804. The subscriber proxy then sets a request timer in step 2806. The subscriber proxy then request work from the access server 524 and transmits a status to the access server 524 in step 2808 before returning to step 2802. If the next event is the expiration of the request timer, the subscriber proxy checks if the request timer expired in step 2810. The subscriber proxy then registers a timeout in step 2812 before returning to step 2806.

If the next event is an access server 524 reply, the subscriber proxy receives the access server 524 reply in step 2814. The subscriber proxy then checks if the reply is a continue in step 2816. If the reply is a continue, the subscriber proxy returns to step 2806. If the reply is not a continue, the subscriber proxy processes the configuration information or action in step 2818 before returning to step 2806. If the next event is a user request for service, the subscriber proxy receives the user request in step 2820. The subscriber proxy then checks if the user is valid in step 2822. If the user is valid, the subscriber proxy then checks if the request is valid in step 2824. If the request is valid, the subscriber proxy packages the request with the security

certification and encryption in step 2826. The subscriber proxy then transmits the request to the appropriate network destination in step 2828 before returning to step 2802. If the user or request is not valid, the subscriber proxy registers a security event in step 2830 before returning to step 2802. The subscriber proxy exchanges user information between the user network 510 and the appropriate network destination.

Dynamic Proxies

One prior system named "pluxy" allows a manual extension of a proxy. Pluxy requires manual determination and loading of a dynamic set of services, which is done on an operator basis. FIGS. 5 and 29 disclose one embodiment for dynamic proxies in an example of the invention. In this embodiment, both the user based proxies and the subscriber based proxies are extended in response to user actions. The dynamic proxy can be modified and enhanced to evolve with new services and/or protocols. The dynamic proxy size is smaller and more efficient by supporting only the logic that is being used. Implementation and development times of new service and protocol are also reduced because new proxies are not required with the new service and/or protocol.

FIG. 29 depicts a flowchart for a dynamic proxy for dynamic proxies in an example of the invention. FIG. 29 begins in step 2900. In step 2902, the dynamic proxy waits for the next event. If the next event is the completion of the initialization, the dynamic proxy checks if the initialization is complete in step 2904. The dynamic proxy then sets a request timer in step 2906. The dynamic proxy then request work from the access server 524 in step 2908 before returning to step 2902. If the next event is the expiration of the request timer, the dynamic proxy checks if the request timer expired in step 2910. The dynamic proxy then registers a timeout in step 2912 before returning to step 2906.

If the next event is an access server 524 reply, the dynamic proxy receives the access server 524 reply in step 2914. The dynamic proxy then checks if the reply is a continue in step 2916. If the reply is a continue, the dynamic proxy returns to step 2906. If the reply is not a continue, the dynamic proxy processes the configuration information in step 2918 before returning to step 2902. If the next event is a user request, the dynamic proxy receives the user request in step 2920. The dynamic proxy then checks if the user and the request are valid in step 2922. If the user and the request are valid, the dynamic proxy checks if there is a new service or protocol requested in step 2924. If there is a new service or protocol, the dynamic proxy proceeds to step 2934. If there is not a new service or protocol, the dynamic proxy packages the request with the security certification and encryption in step 2926. The dynamic proxy then transmits the request to the appropriate network destination in step 2928 before returning to step 2902. If the user or request is not valid, the subscriber proxy registers a security event in step 2930 before returning to step 2902.

If the next event is a new service/protocol request, the dynamic proxy receives the new service/protocol request in step 2932. In step 2934, the dynamic proxy processes the new service/protocol request to generate a handler request for a new service/protocol handler in step 2934 before returning to step 2926. The dynamic proxy receives the handler and executes the handler for the new service or proxy. In one embodiment, the dynamic proxy is enhanced by loading apple-like extensions from a handler system similar to a web browser.

Access Execution Environment

Proxy servers in the Internet represents a user in one network as existing in another network to obtain services for

the user. One problem with Internet proxy servers is the absence of the management of network resource utilization. In TINA-C, the User Agent and the User Session Manager are restricted to known service binding because any new service requires knowledge of or logic for a new CORBA interface. One problem with TINA-C is the absence of any provider resource management capabilities. Unfortunately, both Internet proxy servers and TINA-C do not provide any means of associating or viewing all network resources used by a user access session. Another problem is the lack of securing provider network resource against tampering by accessing users. Also, Internet proxy servers and TINA-C do not provide a platform-supported means of enforcing security levels on network access users. Both system also lack a validation of potential execution capability extensions.

FIGS. 5, 30, and 31 disclose one embodiment for an access execution environment in an example of the invention. The access execution environment easily and efficiently manages and secures network access by providing an execution environment in the access provider environment. Users access shared services and resources through their specific access execution environment. Because the access execution environment is in the access provider environment, the access provider can view the all of the user's activity by observing the execution environment. In this embodiment, the execution environment is setup as a standard system user on the network access platform, which allows management of the system user by operating system level user resource controls, quotas, and management mechanisms.

The access execution environment includes a multi-tasking virtual machine, a script interpreter, alias resolution capabilities, security certificate authentication, configuration handler, session caching, and protocol handling components. FIG. 30 depicts a block diagram of the access execution environment in an example of the invention. In FIG. 30, a network access platform 3000 comprises an execution environment manager 3010, a program and attribute database 3020, an execution environment A 3030, a transport A 3032, an execution environment B 3040, a transport B 3042, an execution environment C 3050, a transport C 3052, an execution environment D 3060, a transport D 3062, an execution environment E 3070, and a transport E 3072. In one embodiment, the network access platform 3000 is included within the database system 522. In another embodiment, the network access platform is included within the host system 575. The transport A 3032 is connected to the execution environment A 3030. The transport B 3042 is connected to the execution environment A 3040 and a workstation 3080. The transport C 3052 is connected to the execution environment C 3050. The transport D 3062 is connected to the execution environment D 3060. The transport E 3072 is connected to the execution environment E 3070. The execution environment A 3030 includes a transport handler 3034, a configuration handler 3036, and a security handler 3038. The execution environment B 3040 includes a transport handler 3044, a configuration handler 3046, and a security handler 3048. The execution environment C 3050 includes a transport handler 3054, a configuration handler 3056, and a security handler 3058. The execution environment D 3060 includes a transport handler 3064, a configuration handler 3066, and a security handler 3068. The execution environment E 3070 includes a transport handler 3074, a configuration handler 3076, and a security handler 3078.

In operation, a system administrator sets up a number of network access user accounts on the hardware platform. The administrator then sets up resource limits for each account/

platform resource. The administrator then starts an execution environment **3030**, **3040**, **3050**, **3060**, and **3070** for each access user account ID. The execution environment **A 3030** initializes and loads a configuration handler **3036**. The execution environment **A 3030** then loads the security handler **3038** and the transport handler **3034**. The transport handler **3034** opens a logical or physical transport port **A 3032** on the hardware platform using port information from the configuration handler.

FIG. 31 depicts a flow chart for the access execution environment in an example of the invention. FIG. 31 begins in step **3100**. In step **3102**, the security handler **3038** waits for a connect message from the transport handler **3034**. The security handler **3038** then receives the connect message from the transport handler **3034** in step **3104**. The security handler **3038** then generates and transmits a logon request via the transport handler **3034** in step **3106**. The security handler **3038** then checks if a reply for the logon request from the transport handler **3034** has been received in step **3108**. If the reply for the logon request has not been received, the security handler **3038** checks if a reply timeout has occurred in step **3110**. If a reply timeout has not occurred, the security handler **3038** returns to step **3108**. If a reply timeout has occurred, the security handler **3038** generates and transmits a disconnect message to the transport handler **3034** in step **3112** and returns to step **3102** to wait for a connect message.

In step **3114**, the security handler **3038** receives the reply for the logon request. Then the security handler **3038** retrieves the location of a security server from the configuration handler **3036** and transmits the logon information to the security server in step **3116**. The security handler **3038** then checks if a reply to the logon information from the security server has been received in step **3118**. If no reply has been received, the security handler **3038** checks if a reply timeout has occurred in step **3120**. If a reply timeout has not occurred, the security handler **3038** returns to step **3118**. If a reply timeout has occurred, the security handler **3038** checks if the logon information has been sent three times in step **3122**. The number of tries could be configurable. If the logon information has not been sent three times, the security handler **3038** returns to step **3116** to transmit the logon information. If the logon information has been sent three times, the security handler **3038** returns to step **3112** to send a disconnect message.

If the security server replied before the reply timeout, the security handler **3038** checks if the reply is an accept message in step **3124**. If the reply is a decline message, the security handler **3038** returns to step **3112**. If the reply is an accept message, the security handler **3038** transmits the accept message configuration parameters to the configuration handler **3036** in step **3126**. The configuration handler **3036** then loads attributes and programs and executes programs specified by the security server reply in step **3128**. The execution environment **A 3030** performs the execution of programs for the user in step **3130**. In another embodiment, the programs request attributes and/or programs to be loaded and/or executed. The programs include the ability to interface with users via the transport handler and load/execute other programs required by request types.

The configuration handler **3036** then checks if the transport handler **3034** receives a disconnect message in step **3132**. If the transport handler **3034** has not received a disconnect message, the configuration handler **3036** returns to step **3130**. If the transport handler **3034** has received a disconnect message, the transport handler **3034** transmits the disconnect message to the configuration handler **3036** in step

3134. In step **3136**, the configuration handler **6536** based on port configuration attributes closes the transport port **3032**, resets the transport handler **3034**, and notifies the execution environment manager **3010** to create a new execution environment for that port. The configuration handler **3036** gracefully shuts down all handlers, programs, and eventually the execution environment **A 3030**. The configuration handler **3036** transmits a shutdown message to the execution environment manager **3010** in step **3138** before terminating in step **3140**. The execution environment manager **3010** restarts the execution environment **A 3030** upon notification of the shutdown. The operations of the other execution environments **3040**, **3050**, **3060**, and **3070** perform in the same manner as the execution environment **A 3030** and are not discussed for the sake of clarity.

Access Network Translations—FIGS. 5, 32, 33, 34, 35, 36, 37, 38, and 39

Domain Name Scoping

The typing in of domain names such as www.nypostonline.com is quite cumbersome for the user to access specific services. The Domain Name Server (DNS) translates the domain name to a network address for the service. Clicking on bookmarks or favorites in browsers, which reference the domain names, does eliminate the need to type in the domain names. However, when users change network devices, these bookmarks or favorites do not follow the user. In an operating system environment, the command interpreter shell allows the user to customize interpretation of command strings. If no customization is loaded, the command interpreters interprets in a default fashion.

FIGS. 5, 32, and 33 disclose one embodiment for domain name scoping in an example of the invention. This embodiment provides a customizable network shell to overlay the standard DNS service offered in Internet Protocol based networks. Users then are able to define user specific acronyms or aliases for specific or logical services. An acronym or alias indicates domain names, macros, programs, actions, or network addresses. If the translation for the alias does not exist in the list of aliases for the user, the access server **524** checks if the alias exists in the list for a group in which the user belongs. The access server **524** then checks if the alias exist in the DNS for the general network. In another embodiment, the database system **522** checks the existence of the alias in the list for the group and in the DNS for the general network. There are numerous variations in the hierarchy for searching for an alias but are not discussed for the sake of simplicity.

FIG. 32 depicts a flowchart for the access server **524** for domain name scoping in an example of the invention. FIG. 32 begins in step **3200**. In step **3202**, the access server **524** waits for a packet. In this embodiment, the access server **524** receives information in the form of packets from the network device **512**. The access server **524** then receives and processes the packet from the network device **512** in step **3204**. The access server **524** then checks if the packet is an alias translation request in step **3206**. If the packet is not an alias translation request, the access server **524** transmits the packet with standard access logic in step **3214** before returning to step **3202**.

If the packet is an alias translation request, the access server **524** then checks if the alias translation is cached for the user in step **3208**. If the alias translation is cached, the access server **524** reformats the packet using the cached alias translation in step **3216** before returning to step **3214**. If the alias translation is not cached, the access server **524** sets an external request timer in step **3210**. The access server **524** then generates and transmits an alias translation request to

the database system 522 in step 3212. The alias translation request is any message, instruction, or signaling indicative of requesting an alias translation. The access server 524 checks whether a reply has been received or the external request timer has expired in step 3218. If a reply has not been received and the external request timer has not expired, the access server 524 returns to step 3218.

If a reply has been received or the external request timer has expired, the access server 524 checks if the reply was valid in step 3220. If the reply is invalid, the access server 524 discards the packet and replies with an invalid alias message or alias error message to the network device 512 before returning to step 3202. If the reply is valid, the access server 524 caches the alias translation for the user in step 3222 before returning to step 3216.

FIG. 33 depicts a flowchart for the database system 522 for domain name scoping in an example of the invention. FIG. 33 begins in step 3300. In step 3302, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 3304. In step 3306, the database system 522 then checks if the access server requester is known in step 3306. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 3308. The database system 522 appends the reply information to the packet in step 3310. The database system 522 replies with a decline message to the access server 524 in step 3312 before returning to step 3302.

If the access server requester is known, the database system 522 then checks whether the request is an alias translation request in step 3314. If the request is not an alias translation request, the database system 522 registers an unknown request event in step 3316 before returning to step 3310. If the request is an alias translation request, the database system 522 identifies and authenticates the user and the network device 512 in step 3318. If the access is invalid, the database system 522 increments a path/device security record in step 3322. The database system 522 then appends the requester denial information to the request packet in step 3324. The database system 522 then replies with a decline message to the access server 524 in step 3312 before returning to step 3302.

If the access is valid, the database system 522 retrieves the user's alias translation list in step 3326. The database system 522 then translates the alias for the user and appends the translation to a reply to the access server 524. The database system 522 then replies with an approve message to the access server 524 in step 3330. In one embodiment of the invention, the database system 522 uses the alias translation system 578 to perform the operations disclosed in FIG. 33 and stores the aliases and alias translations in the alias translation system 578.

Inband Domain Name Service Lookup

The Domain Name Server (DNS) translations of a domain name to a network address delays user interaction with the service provider. In prior solutions in voice telephone Signaling System #7 networks, a signal transfer point performs an inband local number portability (LNP) lookup on a ISDN part call setup request such as an Initial Address Message (IAM). This inband lookup eliminates the network switch from launching a Transaction Capabilities Application Part (TCAP) LNP query to translate the telephone number in the IAM.

FIGS. 5 and 34 disclose one embodiment for inband domain name service in an example of the invention. This embodiment provides a cache for the access server 524 for alias translations on a per user basis. Thus, the external

queries for translations of aliases to domain names, macros, programs, or network addresses are eliminated. If the translated value of a request is in the alias translation cache, no translation request will be required to a DNS server. Therefore, users experience reduced delays when requesting a service.

FIG. 34 depicts a flowchart for the access server 524 for an inband domain name service lookup in an example of the invention. FIG. 34 begins in step 3400. In step 3402, the access server 524 waits for a packet. The access server 524 then receives and processes the packet from the network device 512 in step 3404. The access server 524 then checks if the destination network address and user is valid in step 3406. If the destination network address and user is valid, the access server 524 transmits the packet on the correct path to reach the destination in step 3408 before returning to step 3402.

If the destination network address or the user is invalid, the access server 524 then checks if the alias translation is cached for the user in step 3410. If the alias translation is cached, the access server 524 reformats the packet using the cached alias translation in step 3412 before returning to step 3402. If the alias translation is not cached, the access server 524 sets an external request timer in step 3414. The access server 524 then generates and transmits an alias translation request to the database system 522 in step 3416. The access server 524 checks whether a reply has been received or the external request timer has expired in step 3418. If a reply has been received and the external request timer has not expired, the access server 524 returns to step 3418.

If a reply has been received or the external request timer has expired, the access server 524 checks if the reply was valid in step 3420. If the reply is invalid, the access server 524 discards the packet and replies with an invalid alias message to the network device 512 in step 3424 before returning to step 3402. If the reply is valid, the access server 524 caches the alias translation for the user in step 3422 before returning to step 3412.

Inline Access Service Triggering

Typical firewalls filter packets out on a request by request basis on "what is not allowed". FIGS. 5 and 35 disclose one embodiment for inline access service triggering. In this embodiment, the access server 554 filters packets out on a "what is allowed" basis. Also, the access server 554 performs packet filtering on a service access basis.

FIG. 35 depicts a flowchart for inline access service triggering in an example of the invention. FIG. 35 begins in step 3500. In step 3502, the access server 554 waits for the next packet. In this embodiment, the access server 554 receives information in the form of packets from the network device 512. The access server 554 receives and processes a packet from the network device 512 in step 3504. In step 3506, the access server 554 checks if the protocol, sending address, and the destination address are known and allowed to pass. In one embodiment, the access server 554 checks the database system 522 for the determination made in step 3506. If the protocol, sending address, and the destination address are known and allowed to pass, the access server 554 checks if the sending address is consistent with the path in step 3508. If the sending address is not consistent with the path, the access server 554 registers a path security event in step 3510. The access server 554 then formats the packet with path information in step 3512. The access server 554 also increments the path/device security record and logs the event before returning to step 3502. If the sending address is consistent with the path, the access server 554 transmits the packet on the correct path with standard firewall access in step 3514 before returning to step 3502.

If the protocol, sending address, or the destination address are not known or not allowed to pass, the access server 554 checks if the request is a filter appeal in step 3515. If the request is not a filter appeal, the access server 554 discards the packet in step 3526 before returning to step 3502. If the request is a filter appeal, the access server 554 identifies and authenticates the user and the network device 512 in step 3516. The access server 554 then checks if the access is valid in step 3518. If the access is invalid, the access server 554 returns to step 3510. If the access is valid, the access server 554 retrieves the user access profile and network device's 512 profile in step 3520. The access server 554 then modifies the access logic for filter modification in step 3524. The access server 554 then modifies the protocol and address filters based on the request in step 4824 before returning to step 3514. In some embodiments, the access server 554 modifies the filters in conformance with the user access profile.

Access Service Triggering

Access providers sometimes need to extend their authentication logic beyond their primary access devices. No prior system in data networks extends the authentication logic beyond what is performed by the access provider's access devices. FIGS. 5, 36 and 37 disclose one embodiment for access service triggering. In this embodiment, the access server 524 triggers a request to external access control logic. Access providers extend the access and authentication logic beyond their access devices while maintaining centralized control of the authentication logic and user access profiles. Also, the access server 554 performs packet filtering on a service access basis.

FIG. 36 depicts a flowchart for the access server 524 for access service triggering in an example of the invention. FIG. 36 begins in step 3600. In step 3602, the access server 524 waits for the next packet. The access server 524 then receives and processes a packet from the network device 512 in step 3604. In step 3606, the access server 524 checks if the protocol, sending address, and the destination address are known and allowed to pass. In one embodiment, the access server 524 checks the database system 522 for the determination made in step 3606. If the protocol, sending address, and the destination address are known and allowed to pass, the access server 524 checks if the sending address is consistent with the path in step 3608. If the sending address is not consistent with the path, the access server 524 registers a path security event in step 3610. The access server 524 then formats the packet with path information in step 3612 before returning to step 3602. If the sending address is consistent with the path, the access server 524 transmits the packet on the correct path with standard firewall access in step 3614 before returning to step 3602.

If the protocol, sending address, or the destination address are not known or are not allowed to pass, the access server 524 sets an external request timer in step 3616. The access server 524 then generates and transmits a request with the path information to the database system 522 in step 3618. The access server 524 then checks if a reply has been received or the external request timer has expired in step 3620. If the reply has not been received or the external request timer has not expired, the access server 524 checks if the reply is valid in step 3622. If the reply is invalid, the access server 524 discards the packet in step 3624 before returning to step 3602. If the reply is valid, the access server 524 then modifies the protocol and address filters based on the reply in step 3624 before returning to step 3614.

FIG. 37 depicts a flowchart for the database system 522 for access service triggering in an example of the invention.

FIG. 37 begins in step 3700. In step 3702, the database system 522 waits for the next request. The database system 522 then receives and processes the request in step 3704. In step 3706, the database system 522 then checks if the access server requester is known in step 3706. If the access server requester is not known, the database system 522 registers an unknown requester event in step 3708. The database system 522 then appends the reply to the packet in step 3710. The database system 522 then generates and transmits a decline reply to the access server 524 in step 3712 before returning to step 3702.

If the access server requester is known, the database system 522 then checks if the request is a filter appeal in step 3714. If the request is not a filter appeal, the database system 522 then checks if the request is a security event in step 3716. If the request is not a security event, the database system 522 registers an unknown action event in step 3717 before returning to step 3712. If the request is a security event, the database system 522 increments a path/device security record, appends the requester information to the request packet, and logs the event in step 3718. The database system 522 then checks if the request is a security event in step 3719. If the request is not a security event, the database system 522 then returns to step 3712. If the request is a security event, the database system 522 proceeds to step 3728.

If the request is a filter appeal, the database system 522 then identifies the user and network device 512 in step 3720. The database system 522 then checks if the access is valid in step 3722. If the access is invalid, the database system 522 returns to step 3718. If the access is valid, the database system 522 retrieves the user and network device profiles in step 3724. The database system 522 then generates access logic and appends the access logic to a reply in step 3726. The database system 522 then transmits the approve reply to the access server 524 in step 3728 before returning to step 3702.

Personal URL

The Internet currently uses Uniform Resource Locator (URL) addresses such as www.yahoo.com so that the address is in more human readable form than the network address. Unfortunately, user cannot distinguish themselves by a network address because the user's network address changes when the user's access point changes. FIGS. 5, 38 and 39 disclose one embodiment for a personal URL. In this embodiment, a network user may publish their location on the network by a user alias. A user alias is any alias that relates to the network address of a user. Logically linking a user's current network address with a user alias allows a user to be located wherever the user accesses the network.

FIG. 38 depicts a flowchart for a personal URL lookup in an example of the invention. FIG. 38 begins in step 3800. In step 3802, the database system 522 waits for the next packet. The database system 522 then receives and processes the packet including the user alias. The database system 522 then checks if a user alias translation is cached for this user in step 3806. If the user alias translation is cached, the database system 522 replies with the current network address of the user in step 3808 before returning to step 3802.

If the user alias translation is not cached, the database system 522 sets an external request timer in step 3810. The database system 522 then generates and transmits a request for user alias translation logic in step 3812. In one embodiment, the database system 522 transmits the request for user alias translation logic to the database system that contains the user access profile. The database system 522

then checks whether a reply was received or the external request timer expired in step 3814. If the reply was not received and the external request timer did not expire, the database system 522 returns to step 3814. If a reply was received or the external request timer did expire, the database system 522 checks if the reply was valid in step 3816. If the reply was invalid, the database system 522 replies with an invalid name message in step 3824 before returning to step 3802.

If the reply was valid, the database system 522 checks if there is a current network address for the user in step 3818. If there is no current network address for the user, the database system 522 replies with user alias not currently in network message in step 3820 before returning to step 3802. If there is a current network address for the user, the database system 522 caches the user alias translation in step 3822 before returning to step 3808.

FIG. 39 depicts a flowchart for a personal URL update in an example of the invention. FIG. 39 begins in step 3900. In step 3902, the database system 522 wait for a request. The database system 522 then receives and processes the request to update the user alias in step 3904. In one embodiment, the database system receives the request from a database system that contains the user access profile. In another embodiment, the request comes from the access server 524. The database system 522 then checks if the requester is known in step 3906. If the requester is not known, the database system 522 registers an unknown requester event in step 3910. The database system 522 then transmits a reply with decline in step 3912 before returning to step 3902.

If the requester is known, the database system 522 then checks if the requester is allowed to update the user alias in step 3914. If the requester is not allowed, the database system 522 registers an unauthorized requester event in step 3916 before returning to step 3912. If the requester is allowed to update, the database system 522 then identifies and authenticates the user and current network address in step 3918. The database system 522 then checks if the access is valid in step 3920. If the access is invalid, the database system 522 increments the path/device security record in step 3922. The database system 522 also appends the requester information to the request and logs the event before returning to step 3912.

If the access is valid, the database system 522 then updates the user alias translation with the current network address in step 3924. The database system 522 then replies with an approve in step 3926 before returning to step 3902. In one embodiment of the invention, the database system 522 uses the personal DNS system 579 to perform the operations disclosed in FIGS. 38 and 39.

Access Network Caching—FIGS. 5, 40, 41, 42, 43, and 44

Predictive Caching

Users that are dialed into the network at a rate of 56 kbps or greater typically retrieve information at a rate of 2–4 kbps. Various equipment between the network access provider and the service provider cause this lower rate of transmission. Some examples of the equipment are network access provider equipment, service provider equipment, network backbone overloading, traffic shaping device, firewalls, and routers. One solution for compensating for the lower rate of transmission is caching. Computers typically contain a memory cache for specific application or devices. Firewalls and routers may also contain caches for data. Unfortunately, none of these caches are user specific.

FIGS. 5, 40, 41 and 42 disclose one embodiment of the invention for predictive end user caching. Predictive end

user caching advantageously improves user noticeable delays and slowdowns due to the lower transmission rate. The extension of data requests from the user network 510 beyond the access server 524 is eliminated when the data requested is cached in the database system. Also, using a predictive algorithm reduces delay by improving caching efficiency based on a user's predictable pattern.

FIG. 40 depicts a flowchart for the access server 524 for auditing in an example of the invention. FIG. 40 begins at step 4000. In step 4002, the user network 510 establishes a connection to the access server 524. In step 4004, the access server 524 generates and transmits a login request message to the database system 522. The access server 524 then checks if the database system allowed an access session to be established for the user in step 4006.

If the access session is not established, the access server 524 terminates in step 4020. If the access session is established, the access server 524 then checks if an access session tear down is requested in step 4012. If the access session tear down is requested, the access server 524 generates and transmits a tear down message with user and path information to the database system 522 in step 4014 before terminating in step 4020.

While no session tear down request is received, the access server 524 exchanges packets with the user network 510 depending on the service provided in step 4016. The access server 524 transmits all new packet destinations including the user and path information to the database system 522 in step 4018 before returning to step 4012. In one embodiment, the database system 522 stores the connect request message, the session reject message, the session accept message, the tear down message including the user and path information, and the packets including the user and path information in an audit database system 526. In one embodiment, the connect request message, the session reject message, the session accept message, the tear down message including the user and path information, and the packets including the user and path information are in the form of an audit message. An audit message is any message, information, or signaling that is audited while a user is accessing an access network 520.

FIG. 41 depicts a flowchart for the database system 522 for auditing in user predictive caching in an example of the invention. FIG. 41 begins in step 4100. In step 4102, the database system 522 waits for an audit message. In step 4104, the database system 522 receives and processes the audit message. The database system 522 then checks if the audit message is a session accept message or a tear down message in step 4106. If the audit message is not a session accept message or a tear down message, the database system 522 retrieves the session state object and updates the object with the audit message in step 4112 before returning to step 4102.

If the audit message is a session accept message or a tear down message, the database system 522 then checks if the audit message is a session accept message in step 4108. If the audit message is a session accept message, the database system 522 generates a session state object including the user, device, path, and session ID in step 4114 before returning to step 1802. If the audit message is a tear down message, the database system 522 retrieves the session state object and updates the object in step 4112. Also, the database system 522 stores the event in step 4116. In one embodiment, the database system 522 stores the event in the cache database system 574. The database system 522 removes the active reference from the session state object before returning to step 4102. In one embodiment of the invention, the database system 522 uses the audit database system 573 to perform the operations disclosed in FIG. 41.

FIG. 42 depicts a flowchart for the database system 522 for caching in user predictive caching in an example of the invention. FIG. 42 begins in step 4200. In step 4202, the database system 522 waits for the next event. The database system 522 processes the event in step 4204. In this embodiment, a request for data is a cache request. The database system 522 then checks if the event is a cache request in step 4206.

If the event is a cache request, the database system 522 identifies the user and retrieves the cached user data in step 4208. The database system 522 then checks if the requested data from the cache request is in the cached user data in step 4212. If the requested data is in the cached user data, the database system 522 replies with the requested data in step 4210 before returning to step 4202. If the requested data is not in the cached user data, the database system 522 retrieves the requested data from the service provider via the access server 524 in step 4214. The database system 524 then replies to the user network 510 with the requested data. Alternatively, in another embodiment, the access server 524 transfers the requested data from the service provider to the user network 510 and the database system 522.

In step 4222, the database system 522 updates the user access profile and the caching algorithm. The database system 522 then determines whether there is a change in the cached data based on the caching algorithm in step 4230. If there is no change in the cached data, the database system 522 returns to step 4202. If there is a change in the cached data, the database system 522 sets any new caching timers for the cached data in step 4236. The database system 522 also generates and transmits a translation change to the access server 524 to change the requested destination address to the database system 522 in step 4238 before returning to step 4202.

If the event is not a cache request, the database system 522 then checks if the event is an audit or timer request in step 4216. If the event is an audit or timer request, the database system 522 identifies the user and retrieves the user access profile in step 4218. The database system 522 then checks if the event is an audit request in step 4220. If the event is an audit request, the database system 522 proceeds to step 4222. If the event is not an audit request, the database system 522 refreshes the cached data from the service provider of the cached data based on the caching algorithm in step 4228. The database system 522 then resets the caching timer for the cached data in step 4234 before returning to step 4202.

If the event is not an audit or timer request, the database system 522 checks if the event is a reset event in step 4224. If the event is not a reset event, the database system 522 registers a unknown event received in step 4232 before returning to step 4202. If the event is a reset event, the database system 522 clears the user access profile statistics, caching timers, the user's cached data. In one embodiment, the user's cached data is stored in the cached database system 574 and the database system 522 uses the cached database system 574 to perform the operations disclosed in FIG. 42.

User Controlled Caching

Another solution to reduce user noticeable delays and slowdowns due to the lower transmission rate is user controlled caching. FIGS. 5, 40, 43 and 44 disclose one embodiment of the invention for user controlled caching. A user controls caching by creating a script of commands to cache data prior to the user accessing the network. For example, a user selects financial or local weather forecasts to cache, so when the user logs on to the network the information will be immediately available from the cache. Scripting has typi-

cally been done on general purpose computers. In a general purpose computer, the user sets up a sequenced set of commands to be executed when the script is executed. Unfortunately, this type of scripting has not been performed on a network cache.

In this embodiment of user controlled caching, the operation of the access server 524 is as described in FIG. 40. FIG. 43 depicts a flowchart for the database system 522 for auditing in user controlled caching in an example of the invention. FIG. 43 begins in step 4300. In step 4302, the database system 522 waits for an audit message. In step 4302, the database system 522 receives and processes the audit message. The database system 522 then checks if the audit message is a session accept message or a tear down message in step 4306. If the audit message is not a session accept message or a tear down message, the database system 522 retrieves the session state object and updates the object with the audit message in the cache in step 4312 before returning to step 4302.

If the audit message is a session accept message or a tear down message, the database system 522 then checks if the audit message is a session accept message in step 4308. If the audit message is a session accept message, the database system 522 generates a session state object in the cache including the user, device, path, and session ID in step 4314. The database system 522 then checks if the user has a pre-cache script in step 4318. If the user does not have a pre-cache script, the database system 522 returns to step 4302. If the user has a pre-cache script, the database system 522 generates and transmits a pre-cache instruction set to the access server 524 to change the translation filter to access the database system 522 for destinations in the pre-cache script so that any requests for those destinations are fulfilled from the cache in step 4320. The database system 522 then sets cache refresh timers in step 4322 and returns to step 4302.

If the audit message is a tear down message, the database system 522 retrieves the session state object and updates the object in step 4316. Also, the database system 522 stores the event in step 4316. In one embodiment, the database system 522 stores the session state object and the state in the cache database system 527. The database system 522 removes the active reference from the session state object before returning to step 4302. In one embodiment of the invention, the database system 522 uses the audit database system 573 to perform the operations disclosed in FIG. 43.

FIG. 44 depicts a flowchart for the database system 522 for caching in user controlled caching in an example of the invention. FIG. 44 begins in step 4400. In step 4402, the database system 522 waits for the next event. The database system 522 processes the event in step 4404. In this embodiment, a request for data is a cache request. The database system 522 then checks if the event is a cache request in step 4406.

If the event is a cache request, the database system 522 identifies the user and retrieves the cached user data in step 4408. The database system 522 then checks if the requested data from the cache request is in the cached user data in step 4412. If the requested data is in the cached user data, the database system 522 replies with the requested data in step 4410 before returning to step 4402. If the requested data is not in the cached user data, the database system 522 retrieves the requested data from the service provider via the access server 524 in step 4414. The database system 522 then replies to the user network 510 with the requested data. Alternatively, in another embodiment, the access server 524 transfers the requested data from the service provider to the user network 510 and the database system 522.

In step 4422, the database system 522 updates the user access profile and the caching algorithm. The database system 522 then determines whether there is a change in the cached data based on the caching algorithm in step 4430. If there is no change in the cached data, the database system 522 returns to step 4402. If there is a change in the cached data, the database system 522 sets the new caching timer for the cached data in step 4436. The database system 522 also generates and transmits a translation change to the access server 524 to change the requested destination address to the database system 522 in step 4440 before returning to step 4402.

If the event is not a cache request, the database system 522 then checks if the event is an audit or timer request in step 4416. If the event is an audit or timer request, the database system 522 identifies the user and retrieves the user access profile in step 4418. The database system 522 then checks if the event is an audit request in step 4420. If the event is an audit request, the database system 522 proceeds to step 4422.

If the event is not an audit request, the database system 522 checks if the event is a script timer in step 4428. If the event is not a script timer, the database system 522 refreshes the cached data from the service provider of the cached data based on the caching algorithm in step 4438 before returning to step 4402. If the event is a script timer, the database system 522 executes the user commands or command set specified by the script timer event in step 4434. The database system 522 then resets the script timer for the cached data in step 4442 before returning to step 4402.

If the event is not an audit or timer request, the database system 522 checks if the event is a reset event in step 4424. If the event is not a reset event, the database system 522 registers a unknown event received in step 4432 before returning to step 4402. If the event is a reset event, the database system 522 clears the user access profile statistics, caching timers, the user's cached data. In one embodiment, the user's cached data is stored in the cached database system 574 and the database system 522 uses the cached database system 574 to perform the operations disclosed in FIG. 44.

Access Network Switching— FIGS. 5, 45, 46, and 47

Switching Access by a User

Access between users and service providers vary based on quality and security, which in turn determine the costs of the access. Users typically have a need to switch between types of access depending on the service offered or what stage in the service the user is in. For example, a user browses the amazon.com website for books using a standard Internet access. When purchasing the books, the user needs a more secure Internet access to ensure that the user's credit card number is not stolen by a hacker.

One prior solution for enhanced security is data Virtual Private Networks (VPN). Data VPN's are relatively static constructions which allow the extension of a user network to another location by extending the network over leased lines or shared Internet/Intranet facilities. However, VPN's require service anticipation, planning and expense beyond what the typical Internet and Intranet users possess. VPN's also do not provide dynamic switching between accesses. Another prior solution is Local Area Network emulation (LANE). LANE utilizes ATM transports to extend the reach of the user network. However, most Internet and Intranet users do not possess ATM equipment and expertise, and the backbone network is still IP.

FIGS. 5 and 45 disclose one embodiment for switching access by a user in an example of the invention. A user

selects a different access to a service provider and is switched to the access without the user's connection to their access provider being interrupted. FIG. 45 depicts a flowchart for switching access by a user in an example of the invention. In this embodiment, a user using standard Internet access selects a premiere Internet access during the setup of a service application. There are numerous variations in switching between access paths such as from premiere Internet access path to a lower quality Internet access path, but only one embodiment is shown for the sake of simplicity.

FIG. 45 begins in step 4500. In step 4502, a user through the network device 512 transmits a request using a standard Internet access path through the service network 530 for a service application residing in the network device 562. The service application in the network device 562 receives the request and transmits a query for the quality of service (QOS) Internet access desired by the requester to the network device 512 in step 4504. The service application in the network device 562 then receives and processes a request to switch access to check if the user selected a premium QOS Internet access in step 4516. If the user did not select a premium QOS Internet access, the user and service application exchange packets using the standard Internet access via the network device 512, the access server 524, the service network 530, the access server 554, and the network device 562 in step 4516. The service application in the network device 562 returns the control to the user in step 4514 to select another service application in step 4502.

If the user selects a premium QOS Internet access, the service application in the network device 562 generates and transmits an authentication and authorization instruction for the premium QOS Internet access to the database system 522 to check if the switch of access is allowed. The database system 522 receives and processes the authentication and authorization instruction. The database system 522 then generates and transmits a premium access instruction to establish premium Internet access between the access server 524 and the access server 554 via the service network 540. In one embodiment, the database system 522 transmits the premium access instruction to the service network 540 to establish premium Internet access between the access server 524 and the access server 554. The database system 522 then generates and transmits the premium access instruction to the access server 524 to connect the network device 512 to the service network 540 for the premium Internet access. The database system 522 also generates and transmits the premium access instruction to the access server 554 to connect the network device 562 to the service network 540 for the premium Internet access.

Once the premium Internet access is established, the user and service application exchange packets using the premium Internet access via the network device 512, the access server 524, the service network 540, the access server 554, and the network device 562 in step 4512. The database system 522 then generates and transmits a disconnect instruction to the access server 524, the service network 540, and the access server 554 to tear down the premium Internet access in step 4518. The database system 522 then generates and transmits usage and performance statistics for the service application in the network device 562 in step 4520 before returning to step 4514. In one embodiment, the database system 522 uses a cross connect system 583 to perform the operations disclosed in FIG. 45.

Switching Access by a Service Provider

FIGS. 5 and 46 disclose one embodiment for switching access by a service provider in an example of the invention. The service provider selects a different access to a user and

41

is switched to the access without the user or service provider's connection to their access provider being interrupted. FIG. 46 depicts a flowchart for switching access by a service provider in an example of the invention. In this embodiment, the service provider selects a toll free premiere Internet service for a user using standard Internet access. The service provider pays for the changes in the toll free premium Internet service instead of the user—similar to toll free phone calls.

FIG. 46 begins in step 4600. In step 4602, the service application in the network device 562 waits for a request for a service. The user through the network device 512 transmits a request using a standard Internet access through the service network 530 for the service application in the network device 562 in step 4604. The service application in the network device 562 receives and processes the request. The service application then selects a toll free premiere quality of service (QOS) Internet access to the network device 512 in step 4606. The service application in the network device 562 generates and transmits an authentication instruction to the database system 522 in step 4608. The database system 522 receives and processes the authentication instruction. In step 4612, the database system 522 then generates and transmits a premium access instruction to establish the toll free premium Internet access between the access server 524 and the access server 554 via the service network 540.

Once the toll free premium Internet access is established, the user and service application exchange packets using the premium Internet access via the network device 512, the access server 524, the service network 540, the access server 554, and the network device 562 in step 4612. Once the service is completed, the database system 522 then generates and transmits a disconnect instruction to the access server 524, the service network 540, and the access server 554 to tear down the toll free premium Internet access in step 4614. The database system 522 then generates and transmits usage and performance statistics for the service application in the network device 562 in step 4616. Once the service is completed, the service application in the network device 562 returns the control to the user in step 4618 to select another service application in step 4602. In one embodiment, the database system 522 uses the cross connection system 583 to perform the operations disclosed in FIG. 46. In another embodiment, a third party such as the government performs the switching for surveillance or monitoring purposes.

Dynamic Switching Access

FIGS. 5 and 47 disclose one embodiment for dynamic switching access in an example of the invention. In a prior solution, a single access link from the user network to the access server is linked to a dedicated network in a single access session. However, users need to switch between different networks such as Internet and Intranets without tearing down the existing access session. No other systems allow the access server to be controlled by the data stream the access server is passing. Analog modems monitor the character stream for an escape sequence. The escape sequence notifies the modem that the data is for modem control and not for application data. However, no prior systems have implemented this functionality for a packet data network.

FIG. 47 depicts a flowchart for dynamic switching access in an example of the invention. In this embodiment, a user during an access session may switch networks without having the access session torn down. The access server 524 receives a control instruction from the user to switch from one dedicated service network 530 to another dedicated service network 540. FIG. 47 begins in step 4700. In step

42

4702, the access server 524 waits for a packet. In this embodiment, a request is in the format of a packet. The access server 524 then receives and processes the packet from the network device 512 in step 4704. The access server 524 checks if the packet is encoded for access control in step 4706. If the packet is not encoded for access control, the access server 524 processes the packet using standard access logic in step 4714 before returning to step 4702.

If the packet is encoded for access control, the access server 524 identifies the physical access path characteristics in step 4708. The access server 524 then checks if the access control is allowed in step 4710. If access control is not allowed, the access server 524 registers and logs an illegal access event in step 4716. The access server 524 then discards the packet in step 4718 before returning to step 4702.

If access control is allowed, the access server 524 generates and transmits an access control instruction to the database system 522 in step 4712. The database system 522 then receives and processes the access control instruction. The database system 522 identifies, authenticates, and authorizes the user and the requesting access server using the packet and path in step 4720. The database system 522 then retrieves the user access profile and the network device profile in step 4722. The database system 522 then checks if access control is allowed for the user and the network device based on the user access profile and the network device profile in step 4724. If access is not allowed, the database system 522 proceeds to step 4716.

If access is allowed, the database system 522 generates and transmits an access instruction to the access server 524 to update the access logic to switch to the service network 540 in step 4726. The database system 522 logs the access change in step 4727. In one embodiment, the database system 522 logs the access change in the audit system 572. The database system 522 also generates and transmits a reply with the new access status to the network device 512 in step 4728 before returning to step 4702. In one embodiment, the database system 522 uses the cross connect system 583 to perform the operations disclosed in FIG. 47.

Access Network Destination Control—FIGS. 5, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, and 59

Network devices become unavailable for a various reasons such as busy, failure, or overload. For network failover, when a destination network device is unavailable, network device requests are manually re-routed to a secondary network device. In order to avoid this manual rerouting, one prior art solution is the sharing of a network address between multiple processors in a box or multiple machines in a cluster. However, the requirement of destination and secondary devices residing in the same box or cluster makes this solution unacceptable for large systems.

Another prior art system is Network Address Translation, which translates a private network address to a public network address for a period of time. This system does not provide the capability of a network address to represent multiple failover destinations. Another solution is the Common Object Request Broker Architecture (CORBA), which provides the user a list of potential "logical" network devices to post the network device request. However, the user must implement CORBA interfaces and interactively select back up processes. The transaction context is lost when the primary fails. The user must re-establish the context. Unfortunately, none of these solutions provide an automatic translation from an unavailable destination network device to a secondary network device without operator

or user intervention, where the destination and secondary devices are not within the same box or cluster.

FIGS. 5, 48 and 49 disclose one embodiment for handling network address failover in an example of the invention. In a failover scenario, a destination network device fails or overloads, which makes the destination network device unavailable, and a secondary network device replicates the destination network device to give the user a fault tolerant appearance of the destination network device. Requests for the destination network device are translated to the secondary network device.

FIG. 48 depicts a flowchart for the access server 554 for network address failover in an example of the invention. FIG. 48 begins in step 4800. In step 4802, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 4804. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 4806, the access server 554 detects that the destination network device 562 fails and checks if the destination network address is enabled for failover in step 4806. If the destination network address is not enabled for failover, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 4812 before returning to step 4802.

If the destination network address is enabled for failover, the access server 554 checks if an address translation is active for the destination network device 562 in step 4808. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 4812 before returning to step 4802. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for failover in step 4810. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 4812 before returning to step 4802.

FIG. 49 depicts a flowchart for the database system 522 for network address failover in an example of the invention. FIG. 49 begins in step 4900. In step 4902, the database system 522 waits for a state event message or a timer event from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or a network device timer expires in step 4904. The database system 522 checks if the network device timer expired in step 4906.

If the network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout in step 4908. The database system 522 then retrieves and updates and the secondary network device's 564 profile to indicate that the destination network device 562 is unavailable in step 4910. In step 4912, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer has not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 4914. In step 4916, the database system 522 checks if the destination network device 562 is known. If the destination network device 562 is not known, the database system 522 generates

and transmits an unknown resource event before returning to step 4902. If the destination network device 562 is known, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics in step 4918. The database system 522 also resets the network device timer. The database system 522 then checks if the state event message is a failure or start event in step 4920.

If the state event message is not a failure or start event, the database system 522 checks if the load event threshold for the destination network device 562 is reached in step 4930. If the load event threshold is not reached, the database system 522 returns to step 4902. If the load event threshold is reached, the database system 522 proceeds to step 4926.

If the state event message is a failure or start event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 4922. In step 4924, the database system 522 checks if the state event message is a failure event. If the state event message is not a failure event, the database system 522 generates new active device profiles and then returns to step 4902. If the state event message is a failure event, the database system 522 proceeds to step 4926.

In step 4926, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address in step 4932 before returning to step 4902. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event in step 4928 before returning to step 4902. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 49.

Network Busy Forwarding

In a Public Switched Telephone Network (PSTN), a busy signal is sent to the caller when the called number destination is busy and thus unavailable. The caller can then select alternate actions such as delaying, retrying, or re-routing the request. Unfortunately, the PSTN handling of busy calls has not been applied to network devices in a data network. FIGS. 5, 50 and 51 disclose one embodiment for network busy forwarding in an example of the invention. When a destination network device 562 is busy, an access server 554 forwards the packets to a secondary network device 564. Requests for the destination network device are translated to the secondary network device.

FIG. 50 depicts a flowchart for the access server 554 for network busy forwarding in an example of the invention. FIG. 50 begins in step 5000. In step 5002, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5004. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5006, the access server 554 checks if the destination network device 562 is busy. If the destination network device 562 is not busy, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5012 before returning to step 5002.

If the destination network device 562 is busy, the access server 554 checks if an address translation is active for the destination network device 562 in step 5008. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step

5012 before returning to step 5002. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for failover in step 5010. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 5012 before returning to step 5002.

FIG. 51 depicts a flowchart for the database system 522 for network busy forwarding in an example of the invention. FIG. 51 begins in step 5100. In step 5102, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or the network device timer expired in step 5104. The database system 522 checks if a network device timer expired in step 5106.

If a network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout such as the network device being busy or unavailable in step 5108. The database system 522 then retrieves and updates and the secondary network device's 564 profile to indicate that the destination network device 562 is busy or unavailable in step 5110. The database system 522 also resets the network device timer. In step 5112, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer is not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 5114. In step 5116, the database system 522 checks if the destination network device 562 is known. If the destination network device 562 is not known, the database system 522 generates and transmits an unknown resource event before returning to step 5102. If the destination network device 562 is known, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics in step 5118. The database system 522 then checks if the state event message is a failure/busy or start/available event in step 5120.

If the state event message is not a failure/busy or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5130. If the load event threshold is not reached, the database system 522 returns to step 5102. If the load event threshold is reached, the database system 522 proceeds to step 5126.

If the state event message is a failure/busy or start/available event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 5122. In step 5124, the database system 522 checks if the state event message is a failure/busy event. If the state event message is not a failure/busy event, the database system 522 generates an address translation clear request in step 5132 before returning to step 5102. The address translation clear request adds a new active device profile or clears the existing translation. If the state event message is a failure/busy event, the database system 522 proceeds to step 5126.

In step 5126, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address if a busy

translation exists in the destination network device's 562 profile in step 5134 before returning to step 5102. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5128 before returning to step 5102. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 51.

Busy Flag

FIGS. 5 and 52 disclose one embodiment for a busy flag when the destination network device is busy in an example of the invention. When a destination network device 562 is busy and thus unavailable, an access server 554 replies to the user with a busy flag. FIG. 52 depicts a flowchart for the access server 554 for a busy flag when the destination network device is busy in an example of the invention. In one embodiment, the busy flag is a busy screen in Hyper-Text Markup Language. FIG. 52 begins in step 5200. In step 5204, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5204. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5206, the access server 554 checks if the destination network device 562 is busy. If the destination network device 562 is not busy, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5210 before returning to step 5202. If the destination network device 562 is busy, the access server 554 replies to the user with a busy flag in step 5208 before returning to step 5202.

Timeout

FIGS. 5 and 53 disclose one embodiment for forwarding if the destination network device times out in an example of the invention. In this embodiment, a destination network device 562 receives a request packet from a user but fails to reply to the user within a pre-determined time. The failure to reply within a pre-determined time indicates the destination network device 562 is unavailable. The access server 554 then redirects the packet to a secondary network device 564 to handle forwarding when the destination network device fails to respond with the pre-determined time.

FIG. 53 depicts a flowchart for the access server 554 for forwarding if the destination network device times out in an example of the invention. FIG. 53 begins in step 5300. In step 5302, the access server 554 waits for the next packet or event. The access server 554 then checks whether a reply timeout event is received in step 5304. If a reply timeout event is not received, the access server 554 then receives and processes the packet in step 5306. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5308, the access server 554 checks if an address translation is available for the destination network device 562. If an address translation is not available for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5318 before returning to step 5302. If an address translation is available for the destination network device 562, the access server 554 sets a reply timer and caches the packet for the secondary network device 564. The access server 554 then transmits the packet with standard access logic to the destination network device 564 in step 5318 before returning to step 5302.

In step 5312, the access server 554 receives the reply timeout event and retrieves the cached request. The access server 554 then processes the reply timeout event and the cached request in step 5314. In step 5315, the access server

554 then replaces the packet's destination network address with the cached secondary network address. The access server 554 then transmits the packet to the secondary network device 564 with the standard access logic in step 5318 before returning to step 5302. In one embodiment, upon reply, the access server 554 deletes the cached packet copy.

Scheduled Alias Resolution

Network devices become unavailable for various reasons such as busy, failure, or overload. Also, service provider need to utilize network resources to improve efficiency and avoid network device latency. In voice telephony signaling networks, an SS7 SCP or an automatic call distributor (ACD) distributes calls to end user call centers. The SCP receives a 800 TCAP query and performs a database lookup to translate the 800 telephone number into a destination telephone number. The SCP may use the requester information and the time of day to perform the database lookup. Some problems with this system are SS7 signaling must be used and the system is limited by voice network constraints. Also, the phone numbers only allow numeric numbers for aliases. In data networks, prior art known as Object Request Brokers (ORB) in a Common Object Request Broker Architecture (CORBA) advertise process resources to a requester. The ORB also registers multiple instances of the same type of process resource.

FIGS. 5, 54, and 55 disclose one embodiment for scheduled alias resolution in an example of the invention. In this embodiment, an alias, domain name/local name, network address, or macro for a network resource is translated to another alias for another network resource based on a configurable schedule. Advantageously, load balancing of network devices is improved by scheduling different alias resolutions for different times/conditions. One problem with CORBA is the requester and resources must implement CORBA interfaces. Scheduled alias resolution is compatible with existing data network DNS implementations. Also, the load balancing, security, and failover may be based on the user access profile and the network device state.

FIG. 54 depicts a flowchart for the access server 554 for schedule alias resolution in an example of the invention. FIG. 54 begins in step 5400. In step 5402, the access server 554 waits for the next request. The access server 554 then receives and processes a request from the network device 512 in step 5404. In this embodiment, the information including an alias is in the form of a request. In step 5406, the access server 554 checks if a current translation for the alias exists. If a current translation for the alias exists, the access server 554 translates the alias to the appropriate network address for a network device in user network 560 in step 5408. The access server 554 then replies to the network device 512 in step 5410 before returning to step 5402.

If a current translation for the alias does not exist, the access server 554 checks if a known access profile exists in the database system 522 exists for this user in step 5412. If a known access profile does not exist in the database system 522 for the user, the access server 554 generates and transmits an alias resolution failure message to the network device 512 in step 5414 before returning to step 5402. If a known access profile does exist in the database system 522 for the user, the access server 554 sets an external request timer in step 5416. The access server 554 then generates and transmits an alias translation request to the database system 522 in step 5418. The access server 554 then checks if a reply is received or the external request timer is expired in step 5420. If the reply is not received or the external request timer is not expired, the access server 554 returns to step

5420. If the reply is received or the external request timer is expired, the access server 554 checks if the reply was valid in step 5422. If the reply is invalid, the access server 554 returns to step 5414. If the reply is valid, the access server 554 then caches the alias translation for this user with time to live parameters in step 5424 before returning to step 5408.

FIG. 55 depicts a flow chart for the database system 522 for scheduled alias resolution in an example of the invention. FIG. 55 begins in step 5500. In step 5502, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 554 in step 5504. In step 5506, the database system 522 then checks if the access server requester is known in step 5506. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 5508. The database system 522 appends the reply to the packet and transmits the reply to the access server 554 in step 5510. The database system 522 replies with a decline message to the access server 554 in step 5512 before returning to step 5502.

If the access server requester is known, the database system 522 then checks whether the request is a translation appeal in step 5514. If the request is not a translation appeal, the database system 522 checks if the request is a security event in step 5516. If the request is a security event, the database system 522 increments a path/device security record in step 5522 before returning to step 5512. If the request is not a security event, the database system 522 registers an unknown event in step 5530 before returning to step 5502. If the request is a translation appeal, the database system 522 identifies and authenticates the user and the network device 512 in step 5518. If the access is not valid, the database system 522 returns to step 5522. If the access is valid, the database system 522 retrieves the user and network device's 512 profiles in step 5524. The database system 522 then generates the translation logic and calculates the translation time to live in step 5526. The database system 522 then appends the translation logic and the translation time to a reply to the access server 554. The database system 522 then replies with an approve message to the access server 554 in step 5528.

Forwarding

In a PSTN, a person may forward calls from their original phone number to another phone number where the person may be reached. Unfortunately, the PSTN handling of call forwarding has not been applied to network devices in a data network. FIGS. 5, 56 and 57 disclose one embodiment for destination controlled forwarding in an example of the invention. An access server 554 forwards the packets for a destination network device 562 that is unavailable to a secondary network device 564. Requests for the destination network device 562 are translated to the secondary network device 564.

FIG. 56 depicts a flowchart for the access server 554 for destination controlled forwarding in an example of the invention. FIG. 56 begins in step 5600. In step 5602, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5604. In this embodiment, the access server 554 receives and processes information in the form of a packet. The access server 554 then checks if the packet is a reply in step 5606. If the packet is a reply, the access server 554 retrieves and deletes the cached destination network address in step 5608 before proceeding to step 5614.

If the packet is not a reply, the access server 554 checks if an address translation is active for the destination network

device 562 in step 5610. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5614 before returning to step 5602. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for destination controlled forwarding in step 5612. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 5614 before returning to step 5602.

FIG. 57 depicts a flowchart for the database system 522 for destination controlled forwarding in an example of the invention. FIG. 57 begins in step 5700. In step 5702, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or a network device timer expires in step 5704.

The database system 522 processes the state event message and authenticates the destination network device 562 in step 5706. In step 5708, the database system 522 checks if the access server requester is known. If the access server requester is not known, the database system 522 generates and transmits an unknown requester event before returning to step 5702. If the access server requester is known, the database system 522 retrieves and updates the access server's 554 profile with availability and performance statistics in step 5710. The database system 522 then checks if the state event message is a failure/forward or start/available event in step 5712.

If the state event message is not a failure/forward or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5716. If the load event threshold is not reached, the database system 522 returns to step 5702. If the load event threshold is reached, the database system 522 proceeds to step 5722.

If the state event message is a failure/forward or start/available event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 5718. In step 5720, the database system 522 checks if the state event message is a failure/forward event. If the state event message is not a failure/forward event, the database system 522 generates and transmits an address translation clear request message to the access server 554 in step 5726 before returning to step 5702. If the state event is a start event, then a new availability profile is generated. If the state event message is a failure/busy event, the database system 522 proceeds to step 5722.

In step 5722, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address if a busy translation exists in the destination network device's 562 profile in step 5728 before returning to step 5702. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5724 before returning to step 5702. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 57.

Network Endpoint Availability Management

FIGS. 5, 58, and 59 disclose one embodiment for network endpoint availability management in an example of the invention. The features of failover, busy, busy flag, timeout, and forwarding are combined into this embodiment. FIG. 58 depicts a flowchart for the access server 554 network endpoint availability management in an example of the invention. FIG. 58 begins in step 5800. In step 5802, the access server 554 waits for the next packet or event. The access server 554 then checks whether a reply timeout event is received in step 5804.

If a reply timeout event is not received, the access server 554 then receives a data packet and processes the packet in step 5806. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5808, the access server 554 checks if the packet is a reply. If the packet is a reply, the access server 554 clears the cached request and clears a reply timer in step 5810 before returning to step 5804. If the packet is not a reply, the access server 554 checks if an address translation is active for the destination network device 562 in step 5812. If an address translation is active for the destination network device 562, the access server 554 proceeds to step 5824. If an address translation is not active for the destination network device 562, the access server 554 checks if the destination address is busy. If the destination address is not busy, the access server 554 proceeds to step 5830. If the destination address is not busy, the access server 554 checks if a forward address translation is available in step 5814. If the forward address translation is not available, replies to the user with a busy flag in step 5818 before returning to step 5802. If the forward address translation is available, the access server 554 proceeds to step 5824.

If a reply timeout event is received, the access server 554 receives the reply timeout event and the cached request in step 5820. The access server 554 then processes the reply timeout event and the cached request in step 5822. In step 5824, the access server 554 then replaces the packet's destination network address with the cached secondary network address. The access server 554 then transmits the packet to the secondary network device 564 with the standard access logic in step 5824.

In step 5826, the access server 554 checks if the forward address is busy. If the forward address is busy, the access server 554 returns to step 5818. If the forward address is not busy, the access server 554 sets a reply timer and caches the request in step 5828. In step 5830, the access server 554 then transmits the packet with standard access logic before returning to step 5802.

FIG. 59 depicts a flowchart for the database system 522 for network endpoint availability management in an example of the invention. FIG. 59 begins in step 5900. In step 5902, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives the state event message from the destination network device 562 or the network device timer expired in step 5904. The database system 522 checks if the network device timer expired in step 5906.

If the network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout such as the network device being busy or unavailable in step 5908. The database system 522 then retrieves and updates the secondary network device's 564 profile to indicate that the destination network device 562 is busy or unavailable in step 5910. The database system 522 also resets the network device timer. In step 5912, the database system 522 generates and transmits an

51

address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer has not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 5914. In step 5916, the database system 522 checks if the access server requester is known. If the access server requester is not known, the database system 522 generates and transmits an unknown resource event before returning to step 5902. If the access server requester is known, the database system 522 retrieves and updates the access server's 554 profile with availability, activity logs, and performance statistics in step 5918. The database system 522 then checks if the state event message is a failure, busy, forward or start/available event in step 5920.

If the state event message is not a failure, busy, forward or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5930. If the load event threshold is not reached, the database system 522 returns to step 5102. If the load event threshold is reached, the database system 522 proceeds to step 5926.

If the state event message is a failure, busy, forward, or start/available event, the database system 522 retrieves and updates the destination network device 562 and secondary network device's 564 profile in step 5922. In step 5124, the database system 522 checks if the state event message is a failure, busy, or forward event. If the state event message is not a failure/busy event, the database system 522 generates and transmits an address translation clear request to the access server 554 if a busy translation exists in step 5934 before returning to step 5902. If the state event message is a failure, busy, or forward event, the database system 522 proceeds to step 5926.

In step 5926, the database system 522 checks if a secondary network device 564 is available for forwarding. If a secondary network device 564 is available, the database system 522 checks if the state event message is a failure or busy event in step 5936. If the state event message is a failure or busy event, the database system 522 generates and transmits a busy address translation request message to the access server 554 in step 5932 before returning to step 5902. If the state event message is not a failure or busy event, the database system 522 generates and transmits a forward address translation request message to the access server 554 to change packets' destination network address to the secondary network address in step 5938 before returning to step 5902. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5928 before returning to step 5902. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 59.

Access Network Audit Server—

FIGS. 5, 60, 61, 62, and 63

Service Capability Monitor

FIGS. 5, 60, 61, 62, and 63 disclose one embodiment for service capability monitor in an example of the invention. In a prior system, Java applet technology and Sun's JINI project delivers communication and state mechanisms from the function provider to the function requesters. Java applet technology allows a mechanism for device control without any coordinating control agent. JINI provides a registry which maintains capabilities like a CORBA ORB as a means for primitive control. One problem is that JINI focuses on

52

the distribution of mechanisms instead of the interworking of mechanisms in a standardized way. In this embodiment of service capability control, a network access provider monitors the validity and state of the application level request to a network service provider. The network access provider monitors performance of their network and the service provider's network service requiring an understanding of the other provider's service requested. Access providers such as Internet service providers could mitigate their liability and increase the service's performance by extending their service view to the user base. Specific service environment access points may be monitored and service access rights and restrictions may be enforced. This embodiment provides an interworking mechanism that provides a dynamic interface to a service based network. The monitoring interface must be dynamic because different service have different potential states, different legal and illegal state transitions, and different communication mechanisms.

FIG. 60 depicts a flowchart for a firewall 526 for service capability monitor in an example of the invention. FIG. 60 begins in step 6000. In step 6002, the firewall 526 waits for a packet. The firewall 526 then receives and processes a packet from the network device 512 in step 6004. In step 6006, the firewall 526 performs standard firewall functions. The firewall 526 then copies the packets and transmits the packet to the destination and the database system 522 in step 6008 before returning step 6002.

FIG. 61 depicts a service capability monitor software architecture for a service capability monitor in an example of the invention. A service capability monitor software architecture 6100 comprises a service broker tokenizer process B 6112, a service broker state map process C 6114, a network A controller process A 6116, a service A signal tokenizer process E 6122, a service A state map process F 6124, a service A controller process D 6126, a service B signal tokenizer process H 6132, a service B state map process F 6134, and a service B controller process D 6136. The network A controller process A 6116 includes a service broker session state objects 6118. The service A controller process D 6126 includes a service A session state objects 6128. The service B controller process D 6136 includes a service B session state object 6138.

The network A controller process A 6116 is connected to the service broker tokenizer process B 6112, the service broker state map process C 6114, the service A controller process D 6126, and the service B controller process D 6136. The service A controller process D 6126 is connected to the service A signal tokenizer process E 6122 and the service A state map process F 6124. The service B controller process D 6136 is connected to the service B signal tokenizer process H 6132 and the service B state map process F 6134.

FIG. 62 depicts a flowchart for the network logic for service capability monitor in an example of the invention. FIG. 62 begins in step 6200. In step 6202, the database system 522 waits for the packet. The database system 522 then receives and processes the packet from the firewall 526 in step 6204. The database system 522 then identifies and authorizes the requester in step 6206. The database system 522 checks if the requester is known in step 6208.

If the requester is known, the network A controller process 6116 writes the packet to process B—service broker signal tokenizer process 6112 in step 6210. The process B 6112 then replies with the function token and parameters to the process A 6116 in step 6212. The database system 522 then checks if a service session state object (SSO) 6118 exists for this session in step 6214. If the SSO 6118 does not exist, the database system 522 creates a SSO 6118 for this session and

53

initializes the current state in step 6216. If the SSO 6118 exists, the database system 522 then proceeds to step 6218. In step 6218, process A 6116 writes the state token and current state to process C—service broker state map process 6114 in step 6218. The process C 6114 then replies with a transition token and parameters to the process A 6116 in step 6220. The process A 6116 applies the transition token and the parameters to the SSO 6118 to update the current state and setting actions in step 6222.

In step 6224, the database system 522 checks if there is an action indicated by the SSO 6118. If there is not an action indicated by the SSO 6118, the database system 522 proceeds to step 6228. If there is an action indicated by the SSO 6118, the database system 522 performs the transition action specified by the SSO 6118 in step 6226. In step 6228, the database system 522 updates the session sliding usage accumulators. The database system 522 then checks if there is an action in the SSO 6118 for this usage in step 6230. If there is no action, the database system 522 returns to step 6202. If there is an action in the SSO 6118, the database system 522 performs the usage action specified in the SSO 6118 before returning to step 6202.

If the requester is not known, the database system 522 checks if there is a network A state object in step 6234. If there is a network state object, the database system 522 proceeds to step 6238. If there is no network state object, the database system 522 creates a network A state object and initializes the current state in step 6236. In step 6238, the database system 522 increments network A state object usage for this event. The database system 522 then checks if there is an action in the network A state object for this usage in step 6240. If there is no action in the network A state object for this usage, the database system 522 returns to step 6202. If there is an action in the network A state object for this usage, the database system 522 performs the usage action specified in the network A state object in step 6242 before returning to step 6242.

FIG. 63 depicts a flowchart for the service logic for service capability monitor in an example of the invention. FIG. 63 begins in step 6300. In step 6302, process D service A controller process 6126 waits for the packet. The process D 6126 then receives and processes the packet from the process A network A controller 6116 in step 6304. The process D 6126 then identifies and authorizes the requester in step 6306. The process D 6126 checks if the requester is known in step 6308.

If the requester is known, the process D 6126 writes the request to process E service A signal tokenizer process 6122 in step 6310. The process E 6122 then replies with the state token and parameters to the process D 6126 in step 6312. The database system 522 then checks if a service session state object (SSO) 6128 exists for this session in step 6314. If the SSO 6128 does not exist, the database system 522 creates a SSO 6128 for this session and initializes the current state in step 6316. If the SSO 6118 exists, the database system 522 then proceeds to step 6318. In step 6218, the process D 6126 writes the state token and current state to process F—service state map process 6124 in step 6318. The process F 6124 then replies with a transition token and parameters to the process D 6126 in step 6320. The process D 6126 applies the transition token and the parameters to the SSO 6128 to update the current state and setting actions in step 6322.

In step 6324, the database system 522 checks if there is an action indicated by the SSO 6128. If there is not an action indicated by the SSO 6128, the database system 522 proceeds to step 6328. If there is an action indicated by the SSO

54

6128, the database system 522 performs the transition action specified by the SSO 6128 in step 6326. In step 6328, the database system 522 updates the session sliding usage accumulators. The database system 522 then checks if there is an action in the SSO 6128 for this usage in step 6330. If there is no action, the database system 522 returns to step 6302. If there is an action in the SSO 6128, the database system 522 performs the usage action specified in the SSO 6128 before returning to step 6302.

If the requester is not known, the database system 522 checks if there is a service A state object in step 6334. If there is a service state object, the database system 522 proceeds to step 6338. If there is no service state object, the database system 522 creates a service A state object and initializes the current state in step 6336. In step 6338, the database system 522 increments service A state object usage for this event. The database system 522 then checks if there is an action in the service A state object for this usage in step 6340. If there is no action in the service A state object for this usage, the database system 522 returns to step 6302. If there is an action in the service A state object for this usage, the database system 522 performs the usage action specified in the service A state object in step 6342 before returning to step 6342.

EXAMPLE

The following is one example of the access communication system that integrates many of the features described in the various embodiments of the inventions above. In the example below, a subscriber called Bank has many employees. The Bank has access to the network architecture 500 through an access provider called AccProv1. In this example, a bank employee is retrieving a stock quote from the yahoo.com website and then switches to request a streaming video from the yahoo.com website. Yahoo's web-server is running in the user network 562. Yahoo has access to the network architecture 500 through an access provider called AccProv2.

A system administrator for AccProv1 configures and optionally starts an access execution environment for the subscriber Bank in the local database system 570. The system administrator also starts a subscriber proxy for the Bank that runs in the access execution environment for the Bank. The subscriber proxy includes transport handlers, configuration handlers, and security handlers. The AccProv1 system administrator starts access execution environments for the maximum number of concurrent users in the local database system 570. The system administrator also starts generic proxies for users in the local database system 570. These generic proxies become user proxies after the user access profile is retrieved for the user.

The system administrator for AccProv2 starts an access execution environment for a service for Yahoo in the local database system 590. The system administrator also starts a service proxy for Yahoo's service that runs in the access execution environment for Yahoo. A system administrator for the Bank then sets up the user access profiles for each employee by inheriting attributes for the user access profile from capability classes or groups that the user belongs to. The user access profile includes the network shell for the user.

The Bank employee then logs on to the access server 524 using a user ID. The Bank system administrator set up. Besides user ID, a user may login to the access server 524 using a prepaid account code, a bank card number, an access card account code, or a user ID including a delimiter for user

access mobility. Upon connection, the access server 524 creates a device session in an access execution environment for the user. Then in response to the user login, the database system 522 retrieves the user access profile for the Bank employee. For a prepaid account code, a bank card number, an access card account code, and a user ID with a delimiter, the database system 522 may retrieve the user access profile from a database system external to the local database system 570. Also, the database system 522 may retrieve the user access profile from a database system external to the local database system 570 for global authentication.

Once the user access profile is retrieved, the access server 524 creates a user session in the access execution environment for the user and binds the device session with the user session. The configuration handler of the user proxy includes the network shell retrieved with the user access profile. The access server 524 then generates and transmits a list of available services in a service based directory to the Bank employee based on the user access profile.

The Bank employee then selects a service to check a stock quote for IBM by transmitting a request for service to the access server 524. The request for service may be in the form of a selection from the service based directory or an alias for alias translation for the network shell using domain name scoping or inbound DNS lookup. The access server 524 processes and transfers the request for service to the user proxy for the Bank employee. The user proxy transfers the request to the service proxy for Yahoo. The service proxy processes and transfers the request for service to Yahoo's web server. The database system 522 translates the service request for Yahoo to a private fulfillment address for one of Yahoo's web servers. The access server 554 then determines if a private destination address is available. If the private destination address is unavailable, the access server 554 could perform many actions including forwarding the request to another Yahoo web server or replying with a busy flag. The access server 524 also binds the user session and the device session with the selected service session by exchanging reference ID's.

The Bank employee then requests another stock quote for Oracle. The access server 524 then determines this stock quote was already cached based on the user's predictable pattern of requesting quotes for this company or a group of software companies. These prior requests for stock quotes of software companies by the Bank employee were audited to derive the Bank employee's predictable pattern. Alternatively, the request for the stock quote for Oracle could have been setup by the user in a script of commands to cache the quote in advance.

The Bank employee then requests a streaming video from Yahoo. The access server 524 switches the access for a premiere access based on the request for streaming video from the Bank employee. Alternatively, Yahoo initiates a switch of access for toll-free Internet service to the Bank employee who is a preferential customer. In order to be able to run the streaming video, the database system 522 also inherits user profile information from a class for streaming video user and updates the user access profile with the user profile information. The user proxy transfers the request for streaming video to the subscriber proxy. The subscriber proxy transfers the request to the service proxy. The service proxy then transfers the request to the Yahoo for one of their streaming video servers. Yahoo's streaming video server then provides the streaming video to the Bank employee.

Conclusion

The access network operates as described in response to instructions that are stored on storage media. The instruc-

tions are retrieved and executed by a processor in the access network. Typically, the processor resides in a server or database. Some examples of instructions are software, program code, and firmware. Some examples of storage media are memory devices, tape, disks, integrated circuits, and servers. The instructions are operational when executed by the processor to direct the network to operate in accord with the invention. The term "processor" refers to a single processing device or a group of inter-operational processing devices. Some examples of processors are computers, integrated circuits, and logic circuitry. Those skilled in the art are familiar with instructions, processors, and storage media.

Those skilled in the art will appreciate variations of the above-described embodiments that fall within the scope of the invention. As a result, the invention is not limited to the specific examples and illustrations discussed above, but only by the following claims and their equivalents.

1 claim:

1. A method of operating an access system including an access server to provide access between a user system and a plurality of communication networks that provide services to a user, the method comprising:

receiving a user login into the access server;

processing the user login to determine if the user is allowed access to the access system based on a local database system;

providing access to the access system to the user in response to the determination that the user is allowed access based on the local database system;

generating an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system;

in the second database system, receiving and processing the authorization query to determine whether the user is allowed access;

in the second database system, generating and transmitting an authorization response to the local database system;

receiving and processing the authorization response indicating whether the user is allowed to use the access system from the second database system; and

providing access to the access system to the user in response to the authorization response that allows the user to use the access system.

2. The method of claim 1 further comprising generating and transmitting a login query.

3. The method of claim 1 wherein processing the user login to determine if the user is allowed access to the access system based on a local database system is based on a user access profile.

4. The method of claim 1 further comprising determining if the second database system exists.

5. The method of claim 1 further comprising disconnecting a user's network device from the access server in response to the determination that the user is not allowed to use the access system.

6. The method of claim 1 wherein the login reply includes an access card account code.

7. The method of claim 1 wherein the login reply includes foreign network account information and further comprising identifying the second external database based on the foreign network account information.

8. The method of claim 1 further comprising:

receiving a request for access into a service control point;

57

processing the request for access to determine the destination for the request;

generating and transmitting a reply to route the request to the access server;

9. The method of claim 1 further comprising logging contract and settlement information.

10. An access system for providing access between a user system and a plurality of communication networks that provide services to a user, the access system comprising:

an access server connected to the user system and the plurality of communication networks and configured to receive and transmit a user logon from the user system to a local database system;

the local database system connected to the access server and configured to receive the user logon, process the user logon to determine if the user is allowed access to the access system based on the local database system, provide access to the access system to the user in response to the determination that the user is allowed access based on the local database system, generate an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system, receive and process an authorization response indicating whether the user is allowed to use the access system from the second database system, and provide access to the access system to the user in response to the authorization response that allows the user to use the access system; and

the second database system connected to the local database system and configured to receive and process the authorization query to determine whether the user is allowed access and generate and transmit the authorization response for the local database system.

11. The access system of claim 10 wherein the access server is configured to generate and transmit a logon query.

12. The access system of claim 10 wherein the local database system is configured to process the user logon to determine if the user is allowed access to the access system based on a user access profile.

13. The access system of claim 10 wherein the local database system is configured to determine if the second database system exists.

14. The access system of claim 10 wherein the access server is configured to disconnect a user's network device in response to the determination that the user is not allowed to use the access system.

15. The access system of claim 10 wherein the logon reply includes an access card account code.

16. The access system of claim 10 wherein the logon reply includes foreign network account information and the local database system is configured to identify the second external database based on the foreign network account information.

58

17. The access system of claim 10 further comprising:

a service control point connected to the user system configured to receive a request for access, process the request for access to determine the destination for the request, and generate and transmit a reply to route the request to the access server.

18. The access system of claim 10 wherein the local database system is configured to log contract and settlement information.

19. A software product for providing access between a user system and a plurality of communication networks that provide services to a user, the software product comprising:

database software operational when executed by a processor to direct the processor to receive the user logon, process the user logon to determine if the user is allowed access to an access system based on a local database system, provide access to the access system to the user in response to the determination that the user is allowed access based on the local database system, generate an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system, receive and process an authorization response indicating whether the user is allowed to use the access system from the second database system, and provide access to the access system to the user in response to the authorization response that allows the user to use the access system; and

a software storage medium operational to store the database software.

20. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to process the user logon to determine if the user is allowed access to the access system based on a user access profile.

21. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to determine if the second database system exists.

22. The software product of claim 19 wherein the logon reply includes an access card account code.

23. The software product of claim 19 wherein the logon reply includes foreign network account information and the database software is operational when executed by the processor to direct the processor to identify the second external database based on the foreign network account information.

24. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to log contract and settlement information.

* * * * *



US006784924B2

(12) United States Patent
Ward et al.**(10) Patent No.: US 6,784,924 B2****(45) Date of Patent: Aug. 31, 2004****(54) NETWORK CONFIGURATION FILE FOR
AUTOMATICALLY TRANSMITTING
IMAGES FROM AN ELECTRONIC STILL
CAMERA****(75) Inventors:** Joseph Ward, Hilton, NY (US);
Kenneth A. Parulski, Rochester, NY
(US); James D. Allen, Rochester, NY
(US)**(73) Assignee:** Eastman Kodak Company, Rochester,
NY (US)**(*) Notice:** Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 0 days.**(21) Appl. No.: 09/004,046****(22) Filed: Jan. 7, 1998****(65) Prior Publication Data**

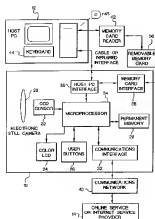
US 2003/0142215 A1 Jul. 31, 2003

Related U.S. Application Data**(60)** Provisional application No. 60/037,963, filed on Feb. 20,
1997, and provisional application No. 60/037,962, filed on
Feb. 20, 1997.**(51) Int. Cl.** **H04N 5/225****(52) U.S. CL.** **348/207.1; 348/211.3****(58) Field of Search** **348/207, 211,**
348/212, 213, 231, 232, 233, 552, 239,
211 99-211.13, 211.3, 17; 710/62, 3, 8;
345/327; 386/48, 117; 725/131-133, 139-141,
151-153, 709/217-219, 220, 705/26, 27,
455/566**(56) References Cited****U.S. PATENT DOCUMENTS**4,524,381 A 6/1985 Konishi
4,574,319 A 3/1986 Konishi
4,825,457 A 4/1989 Lebowitz
5,241,659 A 8/1993 Parulski et al.
5,477,264 A 12/1995 Sarbadhikan et al.5,606,365 A 2/1997 Maurinus et al.
5,663,678 A 9/1997 Chang
5,666,159 A 9/1997 Parulski et al.
5,737,491 A * 4/1998 Allen et al. 704/270
5,800,005 A * 9/1998 Hull et al. 455/566
5,825,432 A * 10/1998 Yonczawa 348/563
6,111,604 A * 8/2000 Hashimoto et al. 348/220
6,122,005 A * 9/2000 Sasaki et al. 348/211.3
6,166,729 A * 12/2000 Acosta et al. 345/719
6,449,001 B1 * 9/2002 Levy et al. 348/1404**OTHER PUBLICATIONS**User's Manual—Axis 2420 Network Camera, 1996 (Axis
Communications).*U.S. patent application Ser. No. 60/037,963, Parulski et al.
“The Universe of Smart Cards: EasySIM software”,
Schlumberger Limited.

“Claris Emailer” from Claris’ home page on internet.

“Kodak Professional Digital Camera System—User’s
Manual,” Eastman Kodak Company, 1991–1992, pp. x to xv,
2–1 to 2–3, 3–29 to 3–30, and 4–1 to 4–49.

* cited by examiner

Primary Examiner—Tuan Ho**(74) Attorney, Agent, or Firm—Pamela R. Crocker****(57) ABSTRACT**A network configuration file is generated at a host computer
and downloaded to a digital camera. This file contains
instruction information for communicating with a selected
destination via a communications interface. The digital
camera includes a “send” button or LCD icon which allows
the user to easily transmit one or more images via a wired
or wireless communications interface to a desired
destination, which among other possibilities may be an
Internet Service Provider or a digital photofinishing center.
When the user selects this option, the communications port
settings, user account specifics, and destination connection
commands are read from the network configuration file on
the removable memory card. Examples of these settings
include serial port baud rate, parity, and stop bits, as well as
account name and password.**9 Claims, 4 Drawing Sheets**

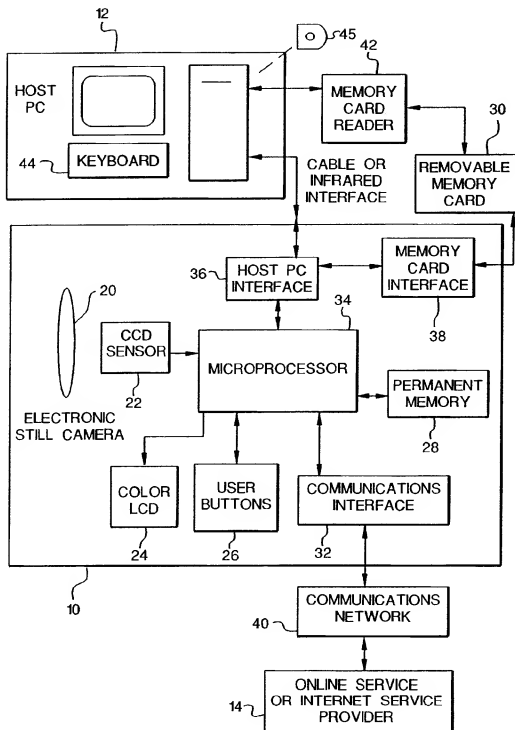
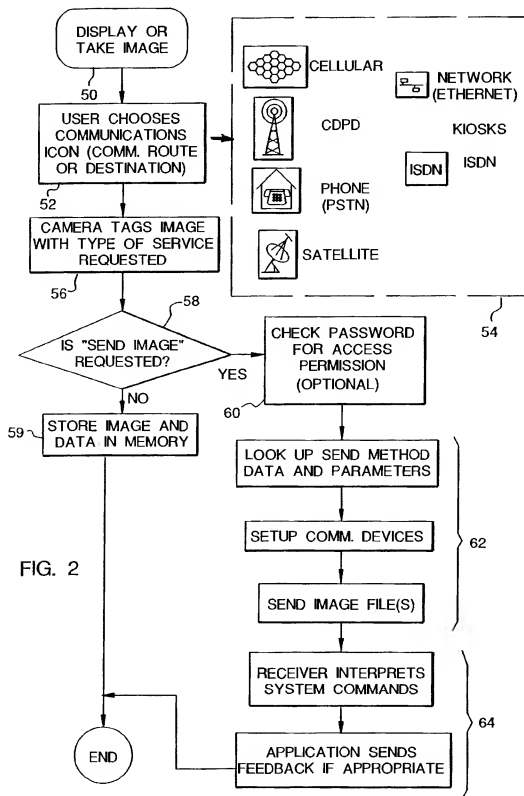


FIG. 1



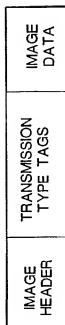


FIG. 3



FIG. 4

FIG. 4A

PHONE (PUBLIC SWITCHED TELEPHONE NETWORK)



PROTOCOL TYPE	PHONE NUMBER	DEFAULT SETTINGS	MODERN CONTROL STRING	ACCOUNT DATA	PASSWORD	SYSTEM COMMANDS
------------------	-----------------	---------------------	-----------------------------	-----------------	----------	--------------------

CELLULAR OR PCS

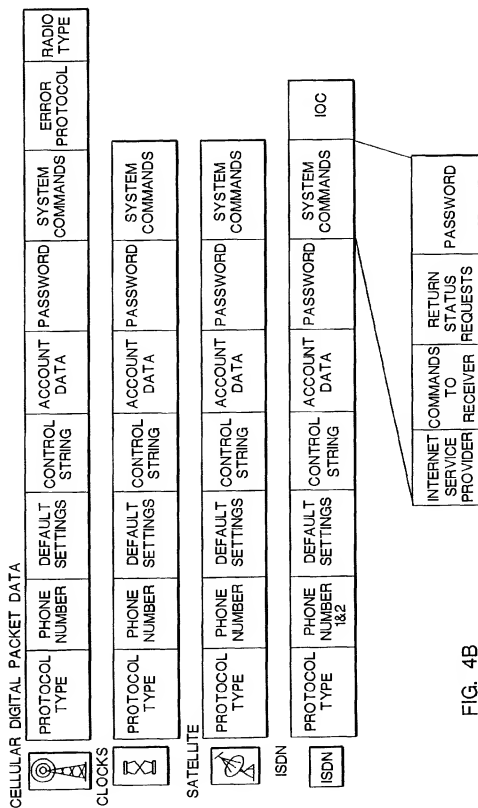


PROTOCOL TYPE	PHONE NUMBER	DEFAULT SETTINGS	MODERN CONTROL STRING	ACCOUNT DATA	PASSWORD	SYSTEM COMMANDS	ERROR PROTOCOL	RADIO TYPE
------------------	-----------------	---------------------	-----------------------------	-----------------	----------	--------------------	-------------------	---------------

WIRELESS LAN



PROTOCOL TYPE	PHONE NUMBER	DEFAULT SETTINGS	PARA- METER FILE	ACCOUNT DATA	PASSWORD	SYSTEM COMMANDS
------------------	-----------------	---------------------	------------------------	-----------------	----------	--------------------



1

NETWORK CONFIGURATION FILE FOR AUTOMATICALLY TRANSMITTING IMAGES FROM AN ELECTRONIC STILL CAMERA

CROSS-REFERENCE TO RELATED APPLICATION(S)

THIS APPLICATION CLAIMS BENEFIT OF PROVI-
SIONAL APPLICATION SERIAL NO. 60/037,962 FILED
Feb. 20, 1997.

Reference is made to commonly assigned copending
applications Serial No. 60/037,963, filed Feb. 20, 1997
entitled "Electronic Camera with 'Utilization' Selection
Capability" and filed in the names of Kenneth A. Parulski,
Joseph Ward, and Michael C. Hopwood, which is assigned
to the assignee of this application.

FIELD OF THE INVENTION

The invention relates generally to the field of
photography, and in particular to electronic photography.
More specifically, the invention relates to a digital camera
that interfaces with a host computer.

BACKGROUND OF THE INVENTION

Digital cameras, such as the Kodak Digital Science
DC25™ camera, allow images to be utilized on a home
computer (PC) and to be incorporated into e-mail documents
and personal home pages on the World Wide Web. Presently,
images must be copied to the PC and transmitted as e-mail,
for example using an online service or an Internet Service
Provider (ISP), via a modem from the user's PC. It would be
desirable to be able to transmit pictures directly from the
digital camera instead of first transferring the pictures to a
PC. For instance, on a vacation trip, it is desirable to
immediately share pictures with friends or relatives via
e-mail or Internet access. It is also desirable to transmit
pictures from a location without PC access in order to free
up camera storage to take additional pictures. There are a
wide variety of connection means to online services such as
America On Line, ISPs, and bulletin board services. Each of
these services typically requires an account name and
password, as well as local telephone access numbers, and
specific communications settings. It would be difficult to
provide an easy-to-use means with buttons or menus on a
small digital camera to input and/or modify all of these
required settings.

SUMMARY OF THE INVENTION

The present invention is directed to overcoming one or
more of the problems set forth above. Briefly summarized,
according to one aspect of the present invention, a network
configuration file is generated at a host computer and
downloaded to a digital camera. This file contains instruction
information for communicating with a selected destina-
tion via a communications interface. The digital camera
includes a "send" button or LCD icon which allows the user
to easily transmit one or more images via a wired or wireless
communications interface to a desired destination, which
among other possibilities may be an Internet Service Pro-
vider or a digital photofinishing center. When the user
selects this option, the communications port settings, user
account specifics, and destination connection commands are
read from the network configuration file. Examples of these
settings include serial port baud rate, parity, and stop bits, as
well as account name and password.

2

In addition, information about which image or images to
transmit is entered using the user buttons on the digital
camera. This information is used to automatically establish
a connection, log-in to the desired destination, and to
transmit the image. The transmission may occur immedi-
ately after the pictures are taken, for example if the camera
has a built-in cellular phone modem, or at a later time, when
the camera is connected to a separate unit (such as a dock,
kiosk, PC, etc.) equipped with a modem. In the latter case,
a "utilization file" is created to provide information on
which images should be transmitted to which account.

These and other aspects, objects, features and advantages
of the present invention will be more clearly understood and
appreciated from a review of the following detailed descrip-
tion of the preferred embodiments and appended claims, and
by reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a system block diagram of the invention.
FIG. 2 is a diagram showing the steps used to automati-
cally transmit images using the network configuration file.
FIG. 3 is a diagram of an image file.
FIG. 4 is a diagram showing several versions of the
network configuration file.

DETAILED DESCRIPTION OF THE INVENTION

Because imaging systems and devices are well known, the
present description will be directed in particular to elements
forming part of, or cooperating more directly with, apparatus
in accordance with the present invention. Elements not
specifically shown or described herein may be selected from
those known in the art. Some aspects of the present descrip-
tion may be implemented in software. Unless otherwise
specified, all software implementation is conventional and
within the ordinary skill in the programming arts.

A system block diagram of the invention is shown in FIG.
1 including an electronic still camera 10, a host computer
(PC) 12 and a service provider 14. The camera includes an
optical section 20 for imaging a scene upon a CCD sensor
22 and generating an image signal, a liquid crystal display
(LCD) 24 for displaying images and other information, a
number of user input buttons 26, both permanent memory 28
and removable memory 30, and an internal communications
interface 32 (e.g., modem). This interface may connect to a
variety of known networks, such as a public switched
telephone network (PSTN), ISDN, an RF cellular phone
network, or Ethernet. The camera 10 also includes a micro-
processor 34 for generally controlling the camera functions,
as well as the interchange of data with the host PC 12 and
the memory card 30 through a host PC interface 36 and a
memory card interface 38, respectively. Besides the host PC
12, the system includes a network connection 40 to the
online service or ISP (Internet Service Provider) 14.
Alternatively, the network 40 can connect to the user's home
PC 12.

When the camera 10 is first purchased (or at any time
thereafter), it is connected to the PC 12 via the host PC 36
interface and a software application (stored on a disc 45)
running on the host PC 12 will enable the user to specify the
name of a destination ISP or online service and to input from
the host PC keyboard 44 the appropriate communication
settings and account information. This information gener-
ates a network configuration file, which then can be then be
downloaded to the camera 10 through the host PC interface

3

4

36, which may be a wired or infrared (e.g., IrDA) interface, and written to the camera's internal memory 28 and/or the removable memory card 30. Alternatively, a host PC equipped with a memory card reader/writer 42 can write the information directly to the card 30 without connecting the camera through its host PC interface 36. Also, this information could be predetermined by the user and stored in a "preferences" file on the host PC 12 and then transferred to the camera 10 from this file without further intervention by the user. Multiple sets of destination services can be stored on the memory card 30. Typically, keyword or graphic descriptors (e.g., icons) accompany the information in the network configuration file about destination services to enable easy access by the camera user.

The steps used to automatically transmit images using the network configuration file are shown in FIG. 2. After disconnecting the camera from the host PC, the user operates the camera to take pictures (step 50). This is typically done at a remote location, for example while traveling to another city. As the user takes or reviews images on the image LCD display, the decision can be made to transmit one or more images (step 52). This is done by choosing one of the keywords or icons in a menu 54 shown in FIG. 2, which are displayed on the LCD 24 and selected, e.g., through the user buttons 26. (Note that a camera will typically only include a subset (only those desired by the user) of all the different services shown.) The selected image files may be tagged with a code (step 56) indicating which service is requested, as shown in FIG. 3. (Alternately, an "image utilization" file can be created in the camera storing a list of images to be transmitted by a particular method, as described in the cross-referenced depending patent application (U.S. Serial No. 60/037,963). As described in that patent application, the details of an order, e.g., number of print copies to be made from an image and the size of the prints and/or a list of images to be e-mailed to various recipients, is written into the "utilization" file, which identifies the order and includes pointers to the image files that store the images required to "fulfill" the order. The "utilization" file is stored in the internal memory 28 or the memory card 30.)

Next, the system determines whether a request exists to send an image (step 58). If no request is present, the image and associated data is stored in either permanent memory 28 or the memory card 30 (step 59). (Typically, all images are initially saved in memory whether eventually sent or not.) Otherwise, if there is a request to send an image, the user ensures that the camera is connected to the appropriate service (wired telephone line, cellular phone, kiosk, etc.) and pushes a "send" button in the user button section 26, or selects a "send" menu option on the LCD 24. The camera then utilizes the appropriate network configuration file, shown in FIG. 4. Each network configuration file contains items such as the protocol type, phone number, etc., as described in Appendix I. The user password may be checked against the password in the network configuration file to ensure that the user is authorized to connect the camera to the desired service (step 60). Alternately, the stored password in the appropriate configuration file can be used. Next, the camera uses the parameters in the configuration file to establish communications with the service and send one or more image files as selected by the user (steps 62). The service receiver interprets the system commands issued by the camera from the network configuration file list and sends appropriate feedback (such as "transfer in progress" and "transfer complete") which are interpreted by the camera and displayed on the LCD 24 (steps 64).

For example, when the camera uses a normal wired telephone (Public Switched Telephone Network) connection

(i.e., network 40) to the camera's internal modem 32, after the user selects the images to be sent and presses the "send" button, the camera performs the following steps without user intervention:

- 1) Read the appropriate connection parameters from the network configuration file (on the memory card 30 or internal camera memory 28), dial the phone and establish the connection to the destination service 14.
- 2) Read the user's account name and password and transmit these to "log-on" to the service 14.
- 3) Using the appropriate communications protocol (FTP, mailto, etc.), transmit the selected image or images to the destination service 14.

The invention has been described with reference to a preferred embodiment. However, it will be appreciated that variations and modifications can be effected by a person of ordinary skill in the art without departing from the scope of the invention.

Appendix I

These are descriptions of the tags listed in the previous drawing:

Protocol Type

Each communication method has its own protocol, or rules to communicate. This tag identifies that protocol and where to find it. For example, the Network may use TCP/IP and a modem may use XModem.

Phone Number

This is the number of the receiving service. If internet access is requested, this could be the number of the Internet Service Provider. For ISDN, some systems require two phone numbers, dialed and connected to in sequence.

Default Settings

Standard settings that make the communications device compatible with the imaging device.

Modem Control String

Modem and communications devices have a command language that can set them up before they are used. For example, modems have many options controlled by command strings including volume level, the amount of time the carrier is allowed to fail before the system hangs up, and so on.

Account Data

This can be internet account data, charge number data, phone card data, billing address, and data related to the commerce part of the transmission.

Password

Any password needed to get into the communications system. Other passwords to get into the remote application or destination are located in the System Commands section.

System Commands

These are commands that control the end destination.

Error Protocol

In cellular and some other wireless communications, error protocols are used to increase the robustness of the link. For example, MNP10 or ETC may be used for cellular links.

Radio Type

The type of radio used for this communications feature may be identified here. Some cell phones have modems built in, others will have protocols for many communications functions built in. The radio type will make the imaging device adapt to the correct interface.

IOC

ISDN Ordering Code identifies what features are available on the ISDN line provided by the telco. It is used to establish the feature set for that communications link.

5

Internet Service Provider

This identifies the actual service provider and any specific information or sequence of information that the service wants to see during connection and logoff. It also tells the device how to handle the return messages, like "time used" that are returned by the server.

Commands to Receiver

This may be a list of commands to control the receiving application. For example, a command to print one of the images and save the data to a particular file on a PC may be embedded here.

Return Status Requests

This tag can set up the ability of the application to tell if an error has occurred, or what the status of the application might be. The data here will help the device decide if it should continue communicating and a set user interface response can be developed around this feedback.

What is claimed is:

1. A method for directing the automatic transmission of images from an electronic camera, said method including the steps of:

storing network configuration information in a memory of the electronic camera;

using the network configuration information to connect the camera to an online service or internet service provider (ISP);

transferring pictures and data from the camera to the online service or ISP;

receiving information from the online service or ISP indicating that the transfer of the pictures and data is complete; and

displaying information on a display of the electronic camera indicating that the transfer of the pictures and data is complete;

wherein the network configuration information is generated at least in part in a host device adaptable for communication with the electronic camera, based on information previously stored in the host device and utilizable to connect the host device to the online service or ISP.

2. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes a PSTN (Public Switched Telephone Network).

3. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes an ISDN network.

4. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes an RF cellular phone network.

6

5. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes an Ethernet connection.

6. The method of claim 1 wherein the information previously stored in the host device comprises information stored in a network configuration preference file of the host device.

7. A method for directing the automatic transmission of images from an electronic camera, said method including the steps of:

storing network configuration information in a memory of the electronic camera, the network configuration information comprising information for using a communications network to establish a connection between the electronic camera and an online service or internet service provider (ISP); and

in response to user activation in the electronic camera of a transmission command specifying one or more pictures for transmission, automatically performing without further user intervention the steps of:

retrieving the stored network configuration information;

utilizing the retrieved information to establish a connection between the electronic camera and the online service or ISP;

transferring the one or more pictures from the electronic camera to the online service or ISP;

receiving information from the online service or ISP indicating that the transfer of the one or more pictures is complete; and

displaying the information on a display of the electronic camera indicating that the transfer of the one or more pictures is complete;

wherein the network configuration information is generated at least in part in a host device based on information previously stored in the host device, and the generated network configuration information is subsequently downloaded from the host device into the electronic camera in conjunction with the storing step.

8. The method of claim 7 wherein the host device comprises a personal computer.

9. The method of claim 7 wherein the information previously stored in the host device comprises information stored in a network configuration preference file of the host device.

* * * * *



US006636259B1

(12) **United States Patent**
Anderson et al.

(10) **Patent No.:** **US 6,636,259 B1**
(45) Date of Patent: **Oct. 21, 2003**

(54) **AUTOMATICALLY CONFIGURING A WEB-ENABLED DIGITAL CAMERA TO ACCESS THE INTERNET**

6,167,469 A * 12/2000 Safai et al. 710/62
 6,226,752 B1 * 5/2001 Gupta et al. 713/201
 6,269,481 B1 * 7/2001 Perlman et al. 717/11
 6,502,195 B1 * 12/2002 Colvin 713/202

(75) Inventors: **Eric C. Anderson**, San Jose, CA (US);
Robert Paul Morris, Raleigh, NC (US)

* cited by examiner

(73) Assignee: **IPAC Acquisition Subsidiary I, LLC**,
 Peterborough, NH (US)

Primary Examiner—Wendy R. Garber
Assistant Examiner—Jacqueline Wilson

(74) *Attorney, Agent, or Firm*—Sawyer Law Group LLP

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 32 days.

(57) **ABSTRACT**

(21) Appl. No.: **09/625,824**

(22) Filed: **Jul. 26, 2000**

(51) **Int. Cl.**⁷ **H04N 5/232**

(52) **U.S. Cl.** **348/211.3; 348/211.12; 348/14.04**

(58) **Field of Search** **348/14.01, 14.02, 348/14.04, 14.08, 14.09, 552, 143, 156, 157, 211.3, 211.12; 709/322**

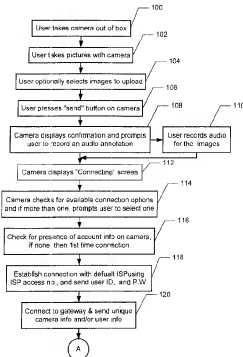
(56) **References Cited**

U.S. PATENT DOCUMENTS

5,430,827 A * 7/1995 Rissanen 704/272
 5,737,491 A * 4/1998 Allen et al. 704/270
 5,905,736 A * 5/1999 Ronen et al. 370/546
 6,064,671 A * 5/2000 Killian 370/389
 6,067,571 A * 5/2000 Igarashi et al. 709/232

The present invention provides a method and system for automatically establishing a user account and for configuring a hand-held electronic device for accessing a site on a public network using the user account. The method and system include establishing a connection to a website server, which is responsible for establishing and maintaining website accounts. The presence of account information on the electronic device camera is then checked, and if none are found, it is determined that this is a first-time connection and information uniquely identifying the electronic device is sent to the website server so that the server can set-up the account information based on the electronic device information. The server then sends user account information to the device, including an account ID and password. The user account information is then stored on electronic device for use the next time the electronic device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network.

22 Claims, 6 Drawing Sheets



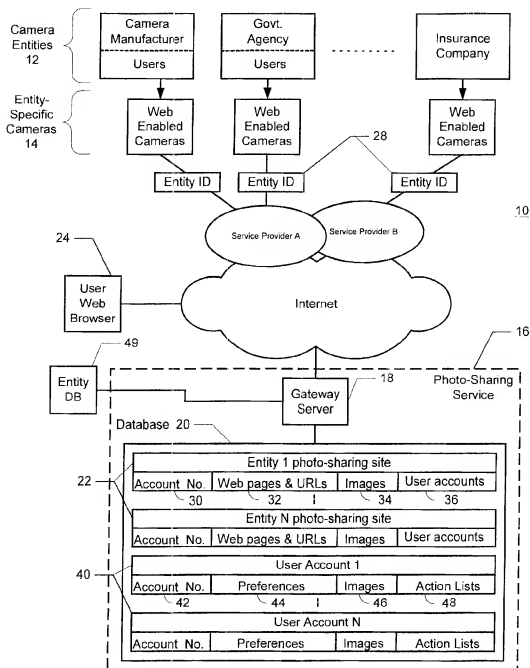


FIG. 1

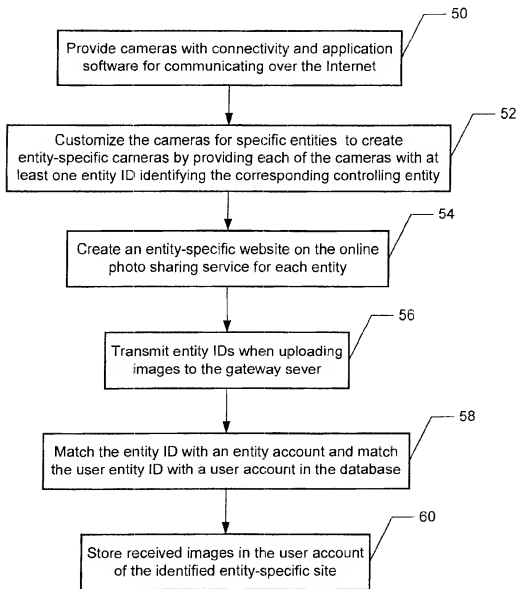


FIG. 2

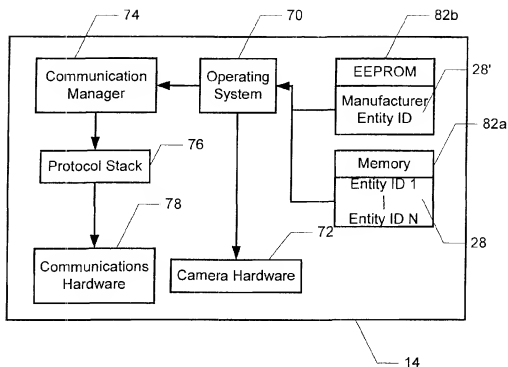


FIG. 3

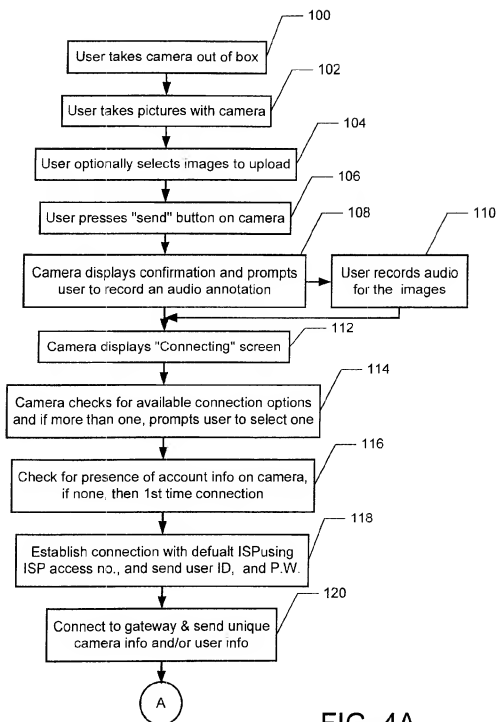


FIG. 4A

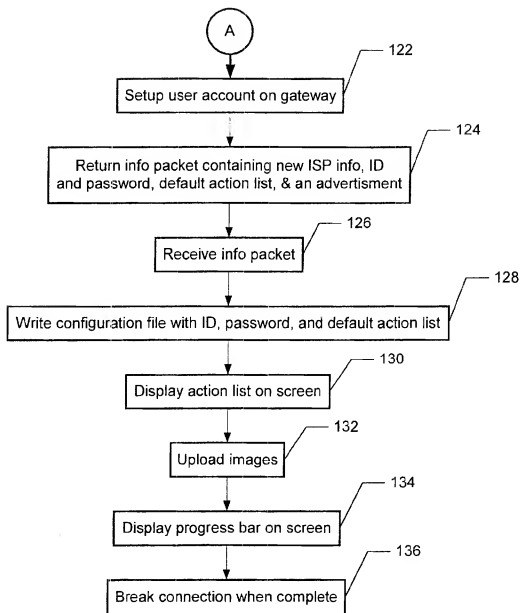


FIG. 4B

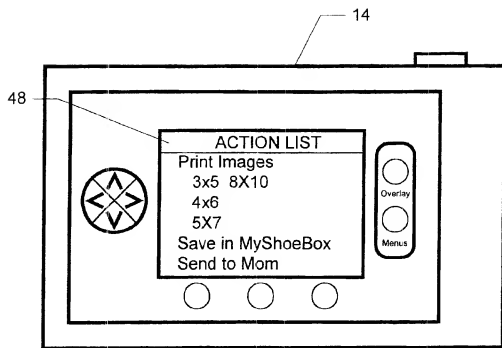


FIG. 5

1

AUTOMATICALLY CONFIGURING A WEB-ENABLED DIGITAL CAMERA TO ACCESS THE INTERNET

CROSS-REFERENCE TO RELATED APPLICATIONS

The present invention is related to co-pending U.S. patent application Ser. No. 09/625,398 entitled "Method and System For Hosting Entity-Specific Photo-Sharing Websites For Entity-Specific Digital Cameras"; and to co-pending U.S. patent application Ser. No. 09/626,418, entitled "Method And System For Selecting Actions To Be Taken By A Server When Uploading Images," which are assigned to the assignee of the present application and filed on the same date as the present application.

FIELD OF THE INVENTION

The present invention relates to a method and system for customizing digital cameras to upload images to an entity-specific photo-sharing websites, and more particularly to automatically configuring a web-enabled digital camera to access the Internet.

BACKGROUND

As the popularity of digital cameras grows, the desire of digital camera users to share their images with others will also continue to grow. New digital camera owners typically try to share their images based on the paradigm of film cameras, in which images are printed on paper and then placed into a photo album. The most straightforward approach to do this with a digital camera is to connect the digital camera directly to a printer to create the prints, and then manually insert the images into a photo album. Users often find this process somewhat complicated and restrictive because standard printers can only print images in limited sizes and requires particular types of paper. And even after the photo album has been assembled, the printed images are not easily shared with many people.

The best approaches to photo-sharing take advantage of the Internet. One such approach is for users to store the digital images on a PC and then send the images to others using email. Several Internet companies now offer an even more convenient approach by providing photo-sharing websites that allow users to store their images for free and to arrange the images into web-based photo albums. Once posted on a photo-sharing website, others may view the images over the Internet.

While convenient for storing digital images, getting the images to the photo-sharing websites can be challenging for users. Most commonly, users must upload their images from the digital camera to a PC using a cable or IrDA, or by inserting the camera's flash card into the PC. From the PC, the user logs onto the Internet and uploads the images to a photo-sharing website. After uploading the images, the user works on the website to arrange the images into web albums and to add any textual information.

Although today's approach for storing images from a digital camera onto a web photo-sharing website and for creating web photo albums works reasonably well, two problems remain that hinder the mainstream adoption of web-based photo-sharing. One problem is that this approach requires the use of a PC, notebook computer, or PDA. While many digital camera users today have PC's, most of those users are early adopters of technology. There are many other consumers who would purchase a digital camera, but are

2

reluctant to do so because they do not yet own a PC or are intimidated by them.

In an effort to address this problem, the assignee of the present application developed an approach to uploading images to the web that doesn't require the use of a PC. In this approach, an email software application is loaded into a digital camera capable of running software that allows the user to e-mail the images directly from the camera. The user simply connects his or her digital camera to a cellphone or modem, runs the e-mail application, and selects the desired images and the email recipients. The selected images are then sent to the recipients as e-mail attachments.

Although emailing photos directly from the camera allows users who do not own a PC to share images over the Internet, these users must still establish accounts with both an Internet service provider (ISP) and the photo-sharing website before being able to post their images. These accounts must also be set-up by PC users as well. For techno savvy users who use a PC to upload the images to the photo-sharing website, establishing the accounts may not be a bother, but even these users may not always have their PC's handy, such as when on vacation, for instance. And for non-PC users, establishing the accounts by entering account information on the digital camera itself may prove to be a cumbersome, if not a daunting, task.

Accordingly, what is needed is an improved method for uploading images from a digital camera to a photo-sharing website on the Internet. In order for online photo-sharing to become more mainstream, an approach that doesn't require a PC or PC expertise and that reduces complexity for the user is required. The present invention addresses such a need.

SUMMARY OF THE INVENTION

The present invention provides a method and system for automatically establishing a user account and for configuring a hand-held electronic device for accessing a site on a public network using the user account. The method and system include establishing a connection to a website server, which is responsible for establishing and maintaining website accounts. The presence of account information on the electronic device camera is then checked, and if none are found, it is determined that this is a first-time connection and information uniquely identifying the electronic device is sent to the website server so that the server can set-up the account information based on the electronic device information. The server then sends user account information to the device, including an account ID and password. The user account information is then stored on electronic device for use the next time the electronic device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network.

According to the method and system disclosed herein, a user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network with the electronic device.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG 1 is a block diagram illustrating an online photo-sharing system in accordance with a preferred embodiment of the present invention.

FIG 2 is a flow chart illustrating a business method for hosting entity-specific photo-sharing websites for entity-specific image capture devices in accordance with a preferred embodiment of the present invention.

3

FIG. 3 is a diagram showing a preferred embodiment of the connectivity and application software of the camera.

FIGS. 4A and 4B illustrate a flow chart of a process for automatically configuring a web-enabled digital camera to access the Internet in a preferred embodiment of the present invention.

FIG. 5 is a block diagram illustrating an example action list 48 displayed on the LCD screen of the camera.

DETAILED DESCRIPTION

The present invention relates to an automatic system for uploading images from a digital camera to entity-specific photo-sharing websites and for automatically establishing accounts. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

FIG. 1 is a block diagram illustrating an online photo-sharing system 10 in accordance with a preferred embodiment of the present invention. The system includes a plurality of camera controlling entities 12 that produce, own, or otherwise control a set of digital cameras 14, and an online photo-sharing service 16. The online photo-sharing service 16 includes a gateway server 18 and an entity/account database 20. The various camera controlling entities 12 contract with the photo-sharing service 16 to transparently host customized photo-sharing websites 22 for each entity, which are referred to herein as entity-specific photo-sharing websites 22. The entity-specific photo-sharing websites 22 are each accessible to the user through the entity's existing Internet site (not shown), and thus appear to users as though the entity-specific photo-sharing websites 22 are hosted by the corresponding entity. According to a preferred embodiment of the present invention, the cameras 14 for a particular entity are customized for that entity to create entity-specific cameras 14, such that when the cameras 14 connect to the Internet, the cameras 14 automatically upload their images to the photo-sharing website of the corresponding entity. In a further aspect of the present invention, the photo-sharing service 16 automatically stores the images in a web album, which is viewable over the Internet by a user's web browser 24.

As used herein, a camera controlling entity 12 is any entity that makes, owns, sells, or controls digital cameras 14, and therefore includes, camera manufacturers, companies, retailers, and end-users. One or more combination of these entities 12 may either contract with the photo-sharing service 16 to provide entity-specific websites 22 for their cameras 14, or have entity information transmitted to the photo-sharing service 16 from the cameras 14. Therefore, a camera controlling entity 12 may include a single entity 12 or a hierarchical relationship of entities 12.

An example of a single entity 12 includes an insurance company that contracts with the photo-sharing service 16 to have all digital cameras 14 used by their agents to transmit their images to a customized insurance photo-sharing website. Examples of a hierarchical relationships of entities 12 includes a camera manufacturer, such as Nikon, that contracts with the photo-sharing service 16 to have all Nikon digital cameras 14 transmit their images to the customized

4

Nikon photo-sharing website. Since the images of different users must be distinguished, each user of a Nikon camera 14 would also constitute an entity within the Nikon website so that the images from different users can be distinguished. Other examples of hierarchical entity relationships include a retailer and its consumers, a real estate agency and its agents, community groups and its members, and government agencies and its employees, for instance.

In a preferred embodiment, the cameras 14 are customized for their respective entities 12 by providing the cameras 14 with software for transmitting entity ID information 28 identifying its controlling entity 12 to the photo-sharing service 16. The photo-sharing service 16 in conjunction with the gateway server 18 and the entity/account database 20 hosts the entity-specific photo-sharing websites 22. Each entity-specific website 22 is identified in the database 20 by an entity account number 30 and includes the web pages and URIs 32 comprising the website, the images and web albums 34 stored on the website, and the user account numbers 36 of authorized users. The database 20 also includes user accounts 40, each of which comprises a user account number 42, user preferences 44, the user's images 46, and action lists 48, explained further below.

The gateway server 18, which communicates with the cameras 14 during image uploading, receives one or more entity IDs 28 from each camera 14 and matches the entity ID 28 with an entity account 30 in the database 20. The images are then automatically associated with the photo-sharing website 22 of the identified entity 12 and/or the identified user.

After the images are uploaded, a user of the camera 14 may visit the online photo-sharing website 22 over the Internet to view the images via a web browser 24. Since the photo-sharing websites 22 are transparently hosted by the photo-sharing service 16, each photo-sharing website 22 appears as though it is hosted by the entity itself, rather than the third party service.

In one embodiment, the cameras 14 may connect to the Internet via a service provider 26, which may include a wireless carrier and/or an Internet service provider (ISP) that is capable of servicing many devices simultaneously. In a preferred embodiment, each of the cameras 14 is provided with wireless connectivity for connecting to the Internet, and are therefore so called "web-enabled" devices, although a wired connection method may also be used.

The cameras 14 may be provided with wireless connectivity using anyone of a variety of methods. For example, a cellphone may be used to provide the digital camera 14 with wireless capability, where the camera 14 is connected to the cellphone via a cable or some short-range wireless communication, such as Bluetooth. Alternatively, the camera 14 could be provided with built-in cellphone-like wireless communication. In an alternative embodiment, the digital camera 14 is not wireless, but instead uses a modem for Internet connectivity. The modem could be external or internal. If external, the camera 14 could be coupled to modem via any of several communications means (e.g., USB, IEEE1394, infrared link, etc.). An internal modem could be implemented directly within the electronics of camera 14 (e.g., via a modem ASIC), or alternatively, as a software-only modem executing on a processor within camera. As such, it should be appreciated that, at the hardware connectivity level, the Internet connection can take several forms. Hence, it should be appreciated that the present invention is not limited to any particular method of accessing the Internet.

5

In a preferred embodiment, the entity-specific websites 22 are customized to seamlessly integrate into the entity's existing website by following the look and feel of the entity's existing website. The entity-specific websites 22 are hosted on the photo-sharing service 16, but a link to the entity-specific websites 22 may be provided on the homepage of the corresponding entity's existing website. Thus, in order to view a web album on an entity-specific website 22, the user must visit the entity's existing website and click the link to the entity-specific website 22, where the user's browser 24 will be transparently directed to the photo-sharing service 16 and be provided with the web pages 32 of the entity-specific website 22.

As an example of the operation of the photo sharing system 10, consider the following scenario. Assume that Minolta and Nikon are entities 12 that have contracted with the photo-sharing service 16, and that the photo-sharing service 16 hosts a photo-sharing website 22 for Minolta and a photo-sharing website 22 Nikon. The Minolta cameras 14 would be provided the entity ID 28 for Minolta and the Nikon cameras 14 would be provided the entity ID 28 for Nikon. When the Minolta and the Nikon cameras 14 send sets of images to the photo-sharing service 16, the gateway server 18 would distinguish the cameras 14 by the entity IDs 28 and would direct the set of images received from Minolta cameras 14 to Minolta's photo-sharing website, and would direct the images from Nikon cameras 14 to Nikon's photo-sharing website. To view the images, the owners of the cameras 14 would use a browser 24 on their PC or PDA to visit the URL of the Minolta or Nikon photo-sharing websites 22. In one preferred embodiment, the photo-sharing service 16 sends the URL of the entity-specific website 22 directly to the camera 14 for display to inform the user of the address.

According to the present invention, the photo-sharing service 16 provides business-to-business and business-to-consumer business models. The service is business-to-business because the service provides companies, such as camera manufacturers, with a complete end-to-end solution for their cameras 14. The solution includes customized software for their cameras 14 for sending images over the internet, and an internet website for storing images from those cameras 14 on a branded website that appears to be hosted by the company. The service is business-to-consumer because it allows users of digital cameras 14 with an automatic solution for uploading captured images from a digital camera 14 to an online photo-sharing website, without a PC.

According to one preferred embodiment of the present invention, the photo-sharing service 16 provides a method of doing business whereby the photo-sharing service 16 shares revenue based on the hierarchical relationship of the entities 12. For example, if the photo sharing service 16 charges a fee for receiving and/or storing the images received from the entity-specific cameras 14, then the photo sharing service 16 may share a portion of the fee with the manufacturer and/or third party supplier of the camera 14 that uploaded the images, for instance. Revenue may also be shared with the wireless service provider providing the connection with the photo sharing service 16.

FIG. 2 is a flow chart illustrating a business method for hosting entity-specific photo-sharing websites for entity-specific image capture devices, such as digital cameras, in accordance with a preferred embodiment of the present invention. First, the cameras 14 are provided with connectivity and application software for communicating over the Internet in step 50. In a preferred embodiment, this step is

6

performed during camera 14 manufacturing to provide off-the-shelf web enabled cameras 14. The cameras 14 are also customized for specific entities 12 to create entity-specific cameras 14 by providing each of the cameras 14 with at least one entity ID 28 identifying the corresponding controlling entity in step 52.

Referring now to FIG. 3, a diagram showing the preferred embodiment of the connectivity and application software of the camera 14 and the entity ID 28 information is shown. Preferably, the camera 14 includes a microprocessor-based architecture that runs an operating system 70 for controlling camera hardware 72 and overall functionality of the camera 14 (e.g., taking pictures, storing pictures, and the like). An example of such an operating system 70 is the Digita™ Operating Environment developed by the assignee of the present application. The camera 14 also includes communication manager 74 software, and a TCP/IP protocol stack 76, that enables communication via the internet, as is well-known in the art. The entity ID information 28 and captured images may be stored in one or more types of memories 82.

For hierarchical entity relationships, the cameras 14 are provided with hierarchical entity IDs 28; one entity ID 28 identifying the entity, and a second entity ID 28 identifying the end-user. Whether there are one or more entity IDs 28, the entity ID 28 of the camera manufacturer may always be provided. Camera 14 customization may occur either during manufacture or anytime thereafter. In a preferred embodiment, the manufacturer entity ID 28 is provided at the time of manufacturing and is stored in an EEPROM 82b, while the entity IDs 28 for other entities 12, such as companies and end-users, are loaded into the camera 14 subsequent to manufacturing and are stored in flash memory 82a or the EEPROM 82b.

Customization that occurs subsequent to manufacture may be implemented using several methods. The first method is to manufacture the cameras 14 with an application programming interface (not shown) for accepting a subsequently loaded software application that specifies the entity ID's 28. The application may come preloaded on a flash card, which is then inserted into the camera 14 by the user and stored in flash memory 82a. The application may also be wirelessly beamed into the camera. When executed in the camera, the software application transmits the appropriate entity IDs 28 to the gateway server 18.

The second method is to load a small file in the camera 14 specifying the entity IDs 28 from a removable memory or from a PC, and storing the file in a system folder within the camera's flash memory 82a. The camera 14 then accesses the file when an Internet connection is established. In a preferred embodiment, the communication manager 74 automatically extracts the manufacturing ID 28 and the entity ID 28 and transmits them to the gateway server 18. In this embodiment, the entity ID 28 is also stored in the EEPROM 82b and is factory set to zero (empty). Thus, unless the entity ID 28 is set, the manufacturing ID 28 may default as the highest controlling entity.

If, for example, a third party developer X contracts to provide custom camera software for camera manufacturer Z, then a custom entity ID will be issued for developer X and developer X will place the custom entity ID into the EEPROM 82b. Developer X is now a controlling entity 12, and may specify to the photo-sharing service 16 that a developer X entity-specific photo-sharing site 22 or developer X's own website be the destination for the uploaded images.

7

The protocol stack 76, under direction of the communications manager interfaces with the communications hardware 78 of camera. The protocol stack 76 includes software APIs and protocol libraries that interface with the communication manager 74, and communication hardware interface drivers that interfaces directly with the various communication hardware 72 the camera 14 must function with (e.g., a Bluetooth transceiver, etc.). The communication manager 74 communicates with operating system 70 and the IP protocol stack 76 to establish an Internet connection ant to transmit the entity ID 28 information and images from the memories 82a and 82b to the photo-sharing service 16.

In an alternative embodiment, rather than loading entity ID's 28 into the camera, a combination of the camera's serial number and the make and model number of the camera may be used as the entity ID 28. Entity specific cameras 14 may then be distinguished by providing a mapping of the camera serial numbers and product IDs to specific entities 12 in the database 20.

Although the camera 14 has been described in terms of a software-based customization solution, those with ordinary skill in the art will readily recognize that the camera 14 may also be provided with a hardware-based solution.

Referring again to FIG. 2, before or after camera customization, the entity-specific websites 22 are created for each entity contracting with the photo-sharing service 16 in step 54. Customization requires storage of entity information in the entity/account database 20 and creating and storing web page elements comprising the entity-specific photo-sharing website in the database 20. The entity-specific information stored in the database 20 may also include service levels, and enabled features for the entity-specific website 22. Features are components or services that may be provided on websites by the photo-sharing service 16, such as search functions, and online printing, for instance, but may be selectable by each entity for its own website. As an example, company X may provide customized cameras 14 for its employees, but may not wish to allow employees to print images from the company X photo website for security reasons. If so desired, company X may have the photo service disable this feature from their particular website.

In a preferred embodiment, the entity-specific websites 22 are not created from scratch, but are created by modifying a preexisting template. The template may include several different sections, such as A, B, C and D, for instance. Assuming for example that the template used to create a website for Nikon, and section A is used to specify the name of the entity then the name Nikon would be inserted into that section. Other entity-specific content would be used to fill out the remaining sections. The Web pages comprising the Nikon specific photo-sharing website would then be provided with URL's unique to that website. The entity's regular website would be modified to include a link to the entity's photo-sharing website 22. In addition, the entity-specific photo-sharing website would include a link back to the entity's website. Entities 12 may have entity photo-sharing websites 22 created for them in one of two ways; automatically by logging into the photo-sharing service 16 and manually customizing the templates, or by having the entity photo-sharing website created for them.

Referring still to FIG. 2, when a camera 14 establishes an internet connection with the gateway server 18, the camera 14 transmits its entity IDs 28 and/or user entity ID 28 when uploading user selected images to the gateway server 18 in step 56. In response, the gateway server 18 matches the entity ID 28 with an entity account in the database 20 and

8

matches the user entity ID 28 with a user account 40 in the database 20 in step 58. The images received are then stored in the user account 40 of the identified entity-specific website 22 in step 60.

Referring again to FIG. 1, each user account 40 in the database 20 may also include one or more action lists 48. According to the present invention, an action list 48 includes one or more items representing actions that the gateway server 18 should take with respect to uploaded images, such as where to store and/or send the images from a particular user or camera, for instance. As explained further below, the action list 48 stored on the database 20 under a user's account 40 are automatically downloaded to the user's camera 14 during a connection with the gateway server 18 and stored on the camera 14. When the user initiates an image upload, the action list 48 is displayed to the user so the user may easily select what actions the gateway server 18 should take with respect to the images by selecting the displayed action list items.

Examples of action list items include specifying that the uploaded images should be stored on the entity-specific photo website, sending the images to a list of email addresses, or even performing some type of analysis or calculation on the image data, for instance.

In a further aspect of the present invention, an action list item is not limited to, instructing the gateway server 18 to perform actions only within the photo-sharing service 16. Rather, an item in the action list 48 may also instruct the gateway server 18 to perform actions outside of the photo-sharing service 16, such as storing the images in an external database 49 of the entity 12. For instance, in the example where the entity 12 is a company, some users of the company's cameras 14 could have action lists 48 instructing the gateway server 18 to store uploaded images to the company's database, rather than to the company's photo-sharing site 22. Based on the action lists 48 and customization, the gateway server 18 may be programmed to automatically perform predefined tasks, such as creating new web albums, or a new page within an existing album, parse the images to extract sound files or other metadata, print images and mail them to designated addresses, and so on.

In a preferred embodiment, the action lists 48 may be created via several methods. In one method, the action list is created by the photo-sharing service 16 the first time the user's camera establishes a connection. That is, a default action list 48 is automatically created based on the entity ID when a user account 40 is first created. In a hierarchical entity relationship where the entity 12 is a company, a default action list 48 may be created to implement a workflow specified by the entity 12. In a hierarchical relationship where the entity 12 is camera manufacturer, for instance, a default action list 48 may be created instructing the gateway server 18 to store the user's images in a simulated "shoebox" on the entity-specific photo-sharing site 20. The user may then go online and create albums from the images in the shoebox as desired.

Another method is for the user to create the action list 48 online on the entity-specific photo-sharing site 20. The action list 48 may be created manually on the website 20 by the user navigating to the site 20 using a web browser 24, accessing her account, and manually creating the action list 48 or editing the action list 48 on the entity-specific site 22. The action list 48 may also be created automatically on the website 20 in response to user actions performed on the website, such as printing images, or creating a web album.

9

Alternatively, after performing an action, the user may be prompted whether they would like this action added to his or her action list 48. If so, the user clicks a check-box and the item is added the action list 48. In a preferred embodiment, any action list 48 created and edited on the photo-sharing site 20 are downloaded to the camera every time the camera 14 connects to the photo-sharing service 16 and made available for user selection on the camera 14 during the next upload.

Yet another method for creating an action list 48 is to allow the user to create the action list on the camera 14. The user may manually create an action list 48 by "typing" in predefined items on the camera. The user may also type in an email address as an action list item whereby when that item is selected, the uploaded images are stored as a web album on the entity-specific photo-sharing website 22 and the server 18 sends a notification to the specified recipient containing the URL to the web album page.

A method and system for hosting web-based photo-sharing websites and for customizing digital cameras to upload images to the entity-specific photo-sharing websites has been disclosed. According to the present invention, users of the customized cameras 14 can upload images to the Internet for storage and web photo album creation without the use of a PC.

In one embodiment described above, the present invention assumes that an ISP account has been established with the digital camera's service provider, and that users of cameras 14 belonging to a certain entity 12 may use the cameras 14 to upload images to the website of the entity 12. However, two problems with account setup remain. One problem is that just as with a PC and PDA, the user must first establish an ISP account before the camera 14 can establish Internet communication. The second account problem is that most websites, including the photo sharing sites 22, may require each user to establish a unique account before using the site to distinguish one user from another. Before being able to connect the web-enabled camera 14 to the Internet, the user must establish these two accounts by either entering account setup information on a PC or entering account setup information on the camera 14. Neither alternative is a convenient alternative for people who do not have the time nor inclination to do so.

In a further aspect of the present invention, the cameras and the photo sharing site are provided with software for automatically creating Internet and photo-sharing website accounts for each camera 14 upon first use, without requiring the user to first enter account information on a PC or on the camera 14.

Referring now to FIGS. 4A and 4B, a flow chart illustrating a process for automatically configuring a web-enabled digital camera to access the Internet in a preferred embodiment of the present invention. Although the process will be described in terms of automatically establishing Internet accounts for a digital camera without requiring the user to enter information, those with ordinary skill in the art will readily recognize that the present invention may be used to automatically establish Internet accounts for any type of portable electronic device.

The process assumes that a user has just acquired a digital camera 14 customized as described above, and has just taken the camera 14 out of its box in step 100. After taking pictures with the camera in step 102, the user may review the images in the camera's LCD screen and optionally select a set of images to upload to the photo sharing service 16 in step 104. The user then presses a "send" button on the camera in step 106 to upload the images.

10

In response, the camera displays a confirmation dialog screen on the camera and prompts the user to record an audio annotation for the images or to continue in step 108. The user may then choose to record audio for the images in step 110. After choosing to continue, or after recording audio, the camera displays a "connecting" dialog screen in step 112 to indicate to the user that the camera is establishing an Internet connection. At the same time, the camera checks for available connection options in step 114, and if more than one is found, the camera prompts the user to select one of the connection options. For example, the camera may be within range of a Bluetooth-equipped printer and a cellphone, so the user will be prompted to choose which device the camera should establish communication with.

The camera then checks for the presence of account information on the camera in step 116, and if there are none, the camera assumes that this is a first-time connection. According to the present invention, in order to allow the camera to make a first-time Internet connection, the camera is provided with default Internet service provider (ISP) information during manufacturing, including an ISP access number, and user ID and password (if required). The camera establishes connection with the default ISP in step 118 by dialing the preloaded access number, and by sending the preloaded user ID and password to the ISP. This special account may be configured so that the camera can only connect to the gateway server 18 (no other IP addresses may be allowed).

After connecting with the ISP, the camera connects to the gateway server 18 and sends unique camera information and/or user information in step 120. In a preferred embodiment of the present invention, a combination of the camera's serial number and the make and model number of the camera may be sent as the unique camera information. In another preferred embodiment, the user's e-mail address may be sent as the unique camera or user information.

Continuing with FIG. 4B, the gateway server 18 uses the unique camera information to set up a user account 40 in step 122. After creating the user account 40, the gateway server 18 returns an information packet to the camera containing new ISP information (if needed), an account ID, and an account password in step 124. The information packet may also contain a default action list specifying what actions should be taken with respect to the images, an advertisement for display on the camera, and the URL of the entity-specific website 22.

It should be noted that if the camera is used in conjunction with an IP direct phone or is provided with a phone number for connected to a dedicated server where the user is not billed separately for the ISP connection, then the steps of providing the camera with default ISP info and returning new ISP info, may be omitted.

The camera receives the information packet in step 126, and writes a configuration file to memory 82a containing the ID, password, and default action list in step 128. The camera then displays the action list on the camera's LCD screen for selection by the user in step 130. The camera may optionally display the user's account information as well.

In an alternative preferred embodiment where security is a concern, the camera 14 first logs off the special ISP and the gateway accounts, and then reconnects using new ISP and gateway accounts in order to retrieve the action lists 48.

FIG. 5 is a block diagram illustrating an example action list 48 displayed on the LCD screen of the camera 14. The action list 48 is shown displaying three major options; printing the uploaded images, saving the uploaded images in

11

the user's shoebox, and sending the images to Mom. Under the printing option, the user may select from various size prints. Below that nested menu categories as shown under the printing option, the action list 48 may be displayed with each action listed as a separate item (e.g., "Send 4x8 prints to Mom", "Send 5x7 prints to me").

Referring again to FIG. 4B, after the user selects one or more actions from the action list 48, the camera begins to upload the images along with the selected actions in step 132 and displays a progress bar on the screen in step 134. In one preferred embodiment, the camera may also display the advertisement sent in the information packet from the gateway server 18. The advertisement may advertise the controlling entity 12, the entity's photo-sharing site 22, or the photo sharing service 16. After all the images are uploaded and associated with the user's account 40, the camera breaks the connection with the gateway server in step 136. At this point, the camera 14 may also display the URL of the entity-specific website 22 to the user.

The next time the user uploads images, the camera will use any new ISP information received to connect to the Internet and will use the account ID and password written to memory 82a when connecting to the gateway server 18. Thus, by using unique camera information, such as the serial number, to establish a web site account upon first use, the present invention eliminates the need for the user to type in information to establish a web site accounts.

To further explain the present invention from a user interaction point of view, consider the following scenario where a user named Jack has just purchased a digital camera 14 from a store and unpacks the camera from the box. There is a "Quick Start Guide" which guides him in getting started. Jack pops in the batteries, sets the date and time, and takes some pictures of his dog and his new baby. Jack can see the pictures have come out well on the small LCD, and now wants to try sharing them with his parents.

The Quick Start Guide says to select the photos to be sent using the "Select" button, and then press the "Send" button. So, Jack navigates to each of the baby pictures, selects them, and then presses the "Send" button. Instantly, a dialog comes up on the LCD screen: "No Receiving Device Found! Please turn on your phone or other connecting device." Oops! Jack pulls out his cell phone, and turns it on. He presses the "Continue" button. Jack does not see this happen, but the camera now "discovers" the cell phone, and immediately presents another dialog: "4 Images Selected. Press Record to add a Sound Note, or Continue to send". Although Jack finds the proposition of recording sound intriguing, he decides to skip it and presses the "Continue" button. Immediately, the dialog is replaced with a "Connecting . . ." dialog.

Shortly, another dialog appears: "Your camera serial number is 38147. Please write this down. You will need it to access your web photo albums". Jack writes the number down on the spot provided in the Quick Start Guide, and presses OK.

Another "Connecting . . ." dialog appears, and is then quickly replaced by another dialog, which says "A free, temporary account has been set up for you at www.photo-sharing.service.com/new_accounts. You will need your camera serial number to access your photos and complete the setup of your account. Please complete the account setup within 30 days". Jack writes down the URL in the space provided in the Quick Start Guide, and presses the OK button. Jack doesn't know it, but during this dialog, the camera has begun transmitting his images and is already partially complete.

12

A new dialog comes up, with a progress bar. Jack is surprised to see that the transmission is already almost ½ done. Below the progress bar, the dialog says "Press 'Continue' to use camera during photo transmission, or wait for progress bar to complete". Jack is interested in watching how fast his images are transmitted, and decides to watch the progress bar complete. A "Transmission Successful" dialog appears. Jack presses the OK button. The camera returns to the review mode.

Jack is pretty excited—he just sent four baby pictures to the Internet. Jack then decides to see what happened to his images so he turns on his PC. After connecting to the Internet, Jack types in the URL from the Quick Start Guide. A Photo-sharing service web page appears, welcoming Jack to the photo-sharing service. After looking briefly at the welcome page, Jack types in his serial number from the Quick Start Guide, and selects his camera's model number from a pop-up menu. Jack clicks on a "Submit" button on the web page.

Jack now sees a page which shows thumbnails of the baby pictures he just sent, the page is entitled "My Shoebox". The page explains to Jack that he is looking at his on-line digital photo shoebox. Since the server 18 knows this is Jack's first visit, special help messages may appear. Various options are provided via buttons and text links. One that catches Jack's eye is CREATE WEB PHOTO ALBUM. Jack clicks this button, and works his way through the process of setting up an on-line photo album. This includes selecting photos from the shoebox, as well as selecting layout and style. One of the check-box items Jack is offered is "Make this album a camera Action List Item". Jack doesn't know what that is, so he clicks the Action List link, which brings up a brief description "If you check this box, you will be able to send pictures directly from the camera to this photo album!" Jack finds this interesting, so he closes the description window, and checks the box. Jack also enters the email address for his parents and his wife's parents, so they can be notified to come and see his photo album, which he has entitled "Our First Baby".

One of the buttons Jack does not click is the "Complete Account Setup" button. He knows that he has 30 days to do that chore, and figures he will get back to it later.

Jack's wife arrives home from shopping at this point, and Jack wants to show her how the new camera works. He starts by showing her the baby album on the PC, then decides to take some pictures of them holding the baby. Jack then selects the images, and presses the "Send" button again.

Since his cell phone is still on, sitting on the table a few feet away, the connection goes smoothly and quickly. A new dialog pops up, surprising Jack. It says "Select the destination for your pictures" and offers two choices: "My Shoebox" and "Our First Baby". Jack is amazed, he doesn't realize that the server 18 downloaded his action list to his camera during the connection. Jack decides to select the "Our First Baby" web album as the destination for the images, and clicks OK. The pictures are sent as before. After the transmission has completed, Jack goes to the PC to check on the photo album. When Jack refreshes the album page, he now sees the additional pictures he just sent, along with the pictures he sent before.

Jack spots a "Send Prints" link on the web page, and clicks it. He is led through a selection of print types, mailing addresses, and credit card info to make it possible to send prints. He is offered the option of completing the setup of his account. Jack decides to do that now, and proceeds to fill out the requested information, including his credit card number.

13

Once the account setup is complete, Jack continues the print order. One of the radio button items is "Make prints a separate Action List Item" or "Make prints part of your Action List". Jack remembers something about Action Lists from before, but is not sure what this means. The description says "Making a separate action list for prints allows you to decide in the camera to send to the photo album and send prints or to just send to the photo album." Jack thinks this is cool, and clicks the "Make prints a separate Action List" item. The next time Jack sends photos from the camera, his action list will be updated to present three choices: My Shoebox, Our First Baby, and Our First Baby w/Prints.

The underlying technology supporting this scenario are summarized below. Other functions and features are assumed, but not required for this scenario:

1. Two-way connection between camera and portal
2. Camera metadata included in the request
3. If not using an IP direct connection,
 - 3.1. Default ISP connection info built into the camera for the first connection (country specific)
 - 3.2. Downloaded assigned ISP information from the portal to the camera
4. Software capable of recognizing automatically a set of supported phones and adjusting the protocol to match
5. Action Lists maintained on the server that are automatically downloaded to the camera to update the camera selection list each time a connection is made
6. An On-line shoebox
7. Ability to download files to the camera from the server. The files could be text, GIF, animated GIF, JPG, or even a script or applet. This feature enables the ability to display advertisements on the camera, remind the user of remaining time to complete account setup, make special offers, and indicate limits reached.

A method and system for automatically configuring a web-enabled digital camera to access the Internet has been disclosed. Although the present invention has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations to the embodiments and those variations would be within the spirit and scope of the present invention. For example, although the photo-sharing service has been described as including the gateway server and the database, the database may be located elsewhere. Also, the gateway server may be used to control account information, while one or more other servers may be used to provide the web pages of the entity-specific websites. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.

What is claimed is:

1. A method for automatically establishing a user account and for configuring a hand-held image capture device which includes a digital camera for accessing a site on a public network using the user account, the method comprising the steps of:

- (a) establishing a connection to a website server, the website server being responsible for establishing and maintaining website accounts;
- (b) checking for the presence of account information on the image Capture Device and if none are found, determining that this is a first-time connection and automatically sending image capture device information uniquely identifying the electronic device to the website server so that the server can set-up the account information based on the image capture device information;

14

(c) receiving user account information from the server, wherein the user account information includes an account ID and password; and

(d) storing the user account information on image capture device for use a next time the image capture device accesses the website, whereby the user does not have to enter account information in order to establish the connection with the public network or to establish the website account.

2. The method of claim 1 further including the step of providing a digital camera as the hand-held image capture device.

3. The method of claim 2 further including the step of sending at least a partial serial number as the image capture device information uniquely identifying the electronic device.

4. The method of claim 3 further including the step of uploading captured images to the website server.

5. The method of claim 4 wherein step (a) further including the steps of:

- (i) providing the image capture device with default ISP information; and
- (ii) establishing a connection to an ISP using the default ISP information.

6. The method of claim 5 further including the step of terminating the connection with the ISP when the uploading of the images is complete.

7. The method of claim 6 further including the step of providing the digital camera with a default ISP access number, user ID, and password.

8. A system for automatically establishing a user account and for configuring a hand-held image capture device which includes a digital camera image capture device for accessing a site on a public network using the user account, comprising:

means for establishing a connection to a website server, the website server being responsible for establishing and maintaining website accounts;

means for checking for the presence of account information on the image capture device, and if none are found, determining that this is a first-time connection and automatically sending image capture device information uniquely identifying the image capture device to the website server so that the server can set-up the account information based on the image capture device information; and

means for receiving user account information from the server, wherein the user account information includes an account ID; and

means for storing the user account information on image capture device for use a next time the image capture device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account.

9. The system of claim 8 wherein the hand-held image capture device comprises a digital camera.

10. The system of claim 9 wherein the image capture device information uniquely identifying the image capture device includes a serial number of the image capture device.

11. The system of claim 10 wherein the camera uploads captured images to the website server.

12. The system of claim 9 wherein the camera further includes means for providing the image capture device with default ISP information, and means for establishing a connection to an ISP using the default ISP information.

13. The system of claim 12 wherein the digital camera terminates the connection with the ISP upon completion of image uploading.

15

14. The system of claim 13 wherein the camera is provided with a default ISP access number, user ID, and password.

15. A method for automatically establishing a user account and for configuring a digital camera to access a website on a public network using the user account, the digital camera including captured images and communication means for accessing the public network, the method comprising the steps of:

- (e) storing default ISP information on the digital camera, the default ISP information including an access number;
- (f) in response to a user request to upload images to the website, determining whether this is a first-time connection;
- (g) if it is first-time connection, establishing communication with the default ISP by dialing the access number;
- (h) establishing communication with the website and automatically sending a digital camera ID to the website;
- (i) using the digital camera ID to set up a user account on the website;
- (j) sending user account information to the digital camera, the user account including an account ID and an account password;
- (k) storing the user account information on the camera; and
- (l) uploading images to the website and associating the images with the user account, whereby the user does not have to enter information to establish a connection with the public network or to establish the website account.

16. The method of claim 15 further including the step of sending at least a partial serial number as the digital camera ID.

17. The method of claim 16 further including the step of receiving new ISP information from the website.

18. The method of claim 17 further including the step of using the new ISP information a next time the digital camera establishes a connection with the public network.

16

19. A system for automatically establishing a user account and for configuring a digital camera for accessing a website on a public network using the user account, the digital camera including images captured by user and communication means for wirelessly accessing the public network, the system comprising the steps of:

- means for storing default ISP information on the digital camera, the default ISP information including an access number;
- means for determining whether this is a first-time connection in response to a user request to upload images to the website;
- means for establishing communication with the default ISP by dialing the access number;
- means for establishing communication with the website and sending a digital camera ID to the website;
- means for using the digital camera ID to set up a user account on the website;
- means for automatically sending user account information to the digital camera, the user account including an account ID and an account password;
- means for storing the user account information on the camera; and
- means for uploading images to the website and associating the images with the user account, whereby the user does not have to enter information to establish a connection with the public network or to establish the website account.

20. The system of claim 19 wherein the digital camera ID includes a serial number.

21. The system of claim 20 wherein the digital camera receives new ISP information from the website.

22. The system of claim 21 wherein the digital camera uses the new ISP information a next time the digital camera establishes a connection with the public network.

* * * * *



US00635384B1

(12) **United States Patent**
Morris

(10) **Patent No.:** **US 6,353,848 B1**
(45) **Date of Patent:** **Mar. 5, 2002**

(54) **METHOD AND SYSTEM ALLOWING A CLIENT COMPUTER TO ACCESS A PORTABLE DIGITAL IMAGE CAPTURE UNIT OVER A NETWORK**

(75) Inventor: **Robert P. Morris**, Raleigh, NC (US)

(73) Assignee: **FlashPoint Technology, Inc.**,
Peterborough, NH (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/127,514**

(22) Filed: **Jul. 31, 1998**

(51) Int. Cl.⁷ **G06F 15/16**

(52) U.S. Cl. **709/203; 709/200; 709/202; 709/217; 709/219; 709/238; 709/232; 709/245**

(58) **Field of Search** **709/200-203, 709/206, 217-219, 227-229, 231-232, 236-237, 245; 707/10, 200-204; 713/201-202**

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,911,044 A	~	6/1999	Lo et al.	709/203
6,018,774 A	~	1/2000	Mayle et al.	709/203
6,058,428 A	~	5/2000	Wang et al.	709/232
6,085,249 A	~	7/2000	Wang et al.	709/229
6,101,536 A	~	8/2000	Kolani et al.	709/217
6,141,759 A	~	10/2000	Braddy	709/203

FOREIGN PATENT DOCUMENTS

DE 198 08 616 9/1998 H04L/12/16

OTHER PUBLICATIONS

De Albuquerque et al., "Remote Monitoring Over The Internet", Nuclear Instruments & Methods In Physics Research, Section-A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 412, No. 1, Jul. 21, 1998, pp. 140-145.

* cited by examiner

Primary Examiner—Zarni Maung

Assistant Examiner—Bharat Barot

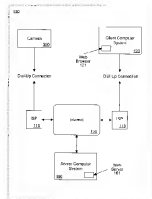
(74) Attorney, Agent, or Firm—Sawyer Law Group LLP

(57)

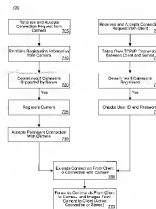
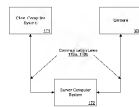
ABSTRACT

A method for accessing a digital image capture unit via a communication network comprising a server computer system and a client computer system communicatively coupled with communication equipment. In one embodiment, the address of the digital image capture unit is registered in an executable program on the server computer system. The executable program is accessed by the client computer system. The executable program connects the digital image capture unit and the server computer system. The executable program communicates commands between the client computer system and the digital image capture unit, such that data captured by the digital image capture unit is transferred to the client computer system via the server computer system.

27 Claims, 11 Drawing Sheets



126



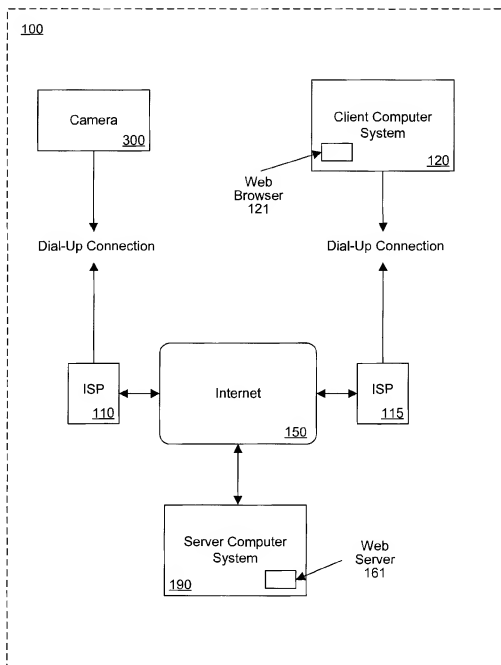


FIG. 1A

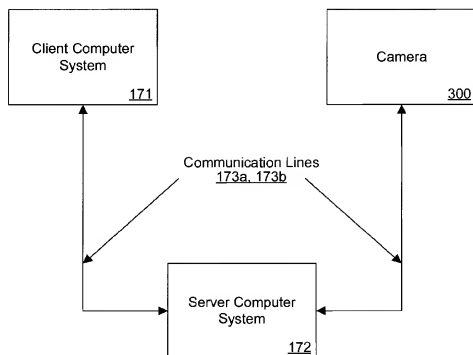
170

FIG. 1B

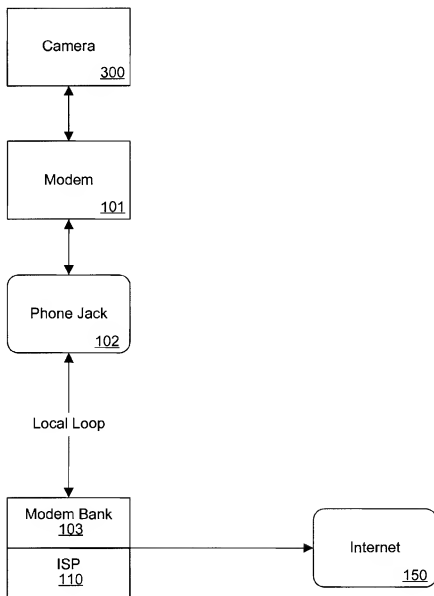


FIG. 1C

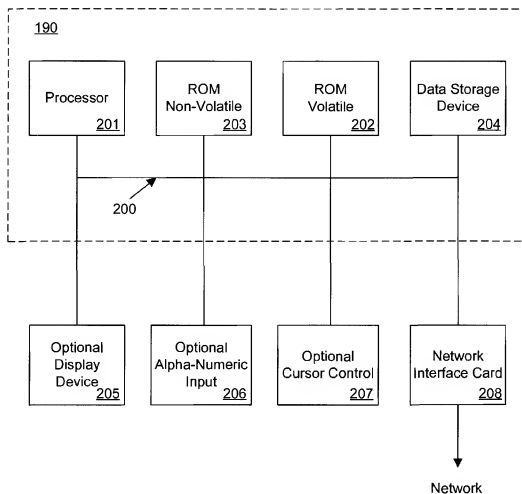


FIG. 2

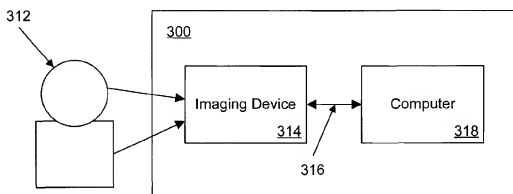


FIG. 3

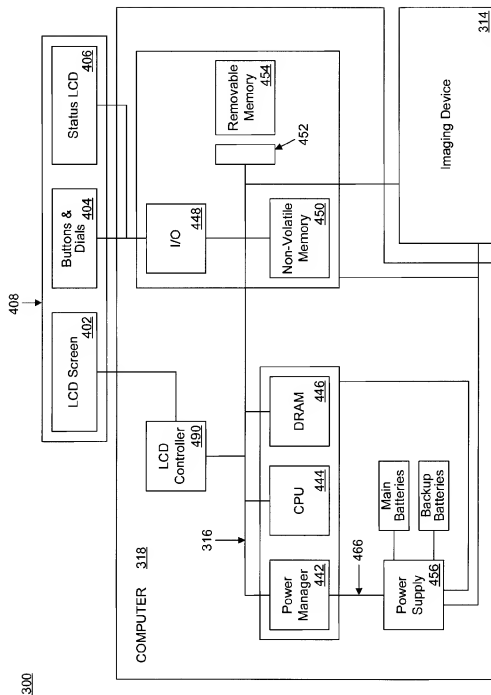


FIG. 4

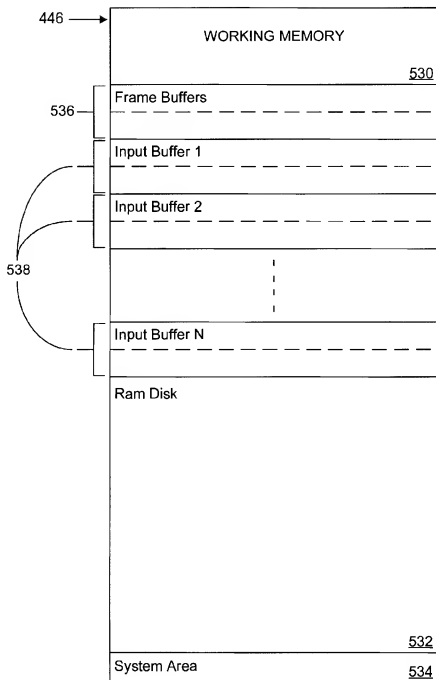


FIG. 5

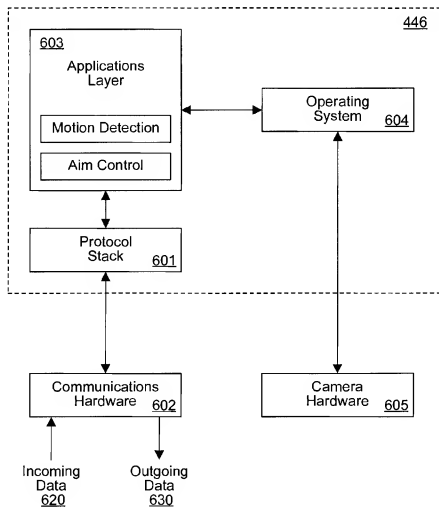


FIG. 6

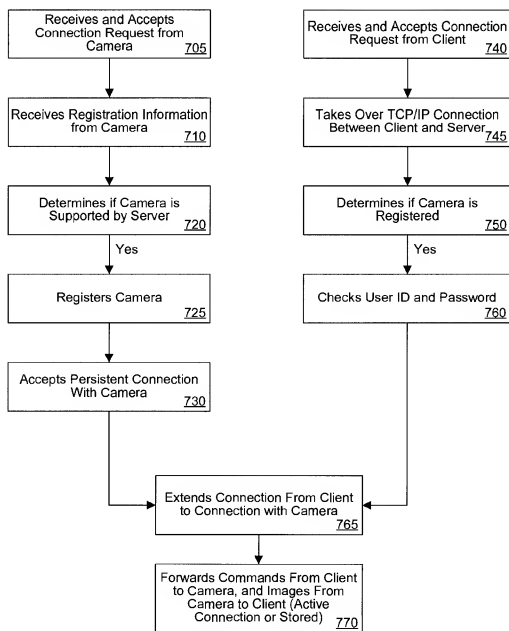
700

FIG. 7

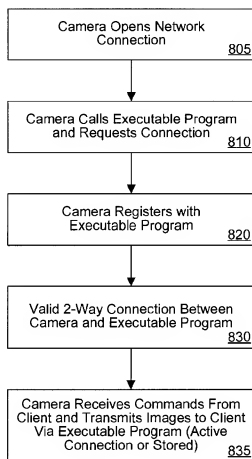
800

FIG. 8

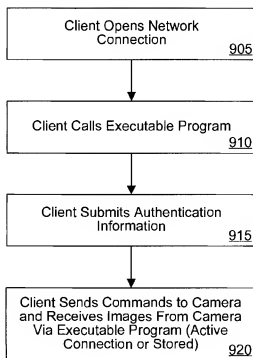
900

FIG. 9

METHOD AND SYSTEM ALLOWING A CLIENT COMPUTER TO ACCESS A PORTABLE DIGITAL IMAGE CAPTURE UNIT OVER A NETWORK

FIELD OF THE INVENTION

The field of the present invention pertains to digital image capture devices. More particularly, the present invention relates to a method for remotely accessing a digital camera via a communication network.

BACKGROUND OF THE INVENTION

Modern digital cameras typically include an imaging device which is controlled by a computer system running a software program. When an image is captured, the imaging device is exposed to light and generates raw image data representing the image. The raw image data are typically stored in an image buffer, where they are processed and compressed by the computer system's processor. Many types of compression schemes can be used to compress the image data, such as the joint photographic expert group (JPEG) standard. After the processor processes and compresses the raw image data into image files, the processor stores the image files in an internal memory or on an external memory card.

Some digital cameras are also equipped with a liquid-crystal display (LCD) or another type of display screen on the back of the camera. Through the use of the LCD, the processor can cause the digital camera to operate in one of two modes, play and record, although some cameras only have a record mode. In the play mode, the LCD is used as a playback screen allowing the user to review previously captured images either individually or in arrays of four, nine, or sixteen images. In the record mode, the LCD is used as a viewfinder through which the user may view an object or scene before taking a picture.

Besides the LCD, user interfaces for digital cameras also include a number of buttons or switches for setting the camera into one of the two modes and for navigating between images in play mode. For example, most digital cameras include two buttons, labeled "-", and "+", that enable a user to navigate or scroll through captured images. For example, if the user is reviewing images individually, then pressing one of navigation buttons causes the currently displayed image to be replaced by the next image.

A digital camera has no film and, as such, there is no incremental cost of taking and storing pictures. Hence, it is possible to take an unlimited number of pictures, wherein the most recent picture replaces the earliest picture, for virtually zero incremental cost. Accordingly, this advantage is best realized when the camera is used as much as possible, taking pictures of practically anything of interest.

One way to best utilize this unique attribute is to make the digital camera and its internally stored images remotely accessible. If the pictures are remotely accessible, the camera could be set to continuously take pictures of scenes and items of interest. Ideally, a user would be able to access those pictures at any time. The user would be able to use a widely available communications medium to access the camera from virtually an unlimited number of locations.

The emergence of the Internet as a distributed, widely accessible communications medium provides a convenient avenue for implementing remote accessibility. Providing remote accessibility via the Internet leverages the fact that the Internet is becoming familiar to an increasing number of

people. Many users have become accustomed to retrieving information from remotely located systems via the Internet. There are many and varied applications which presently use the Internet to provide remote access or remote connectivity. Internet telephony is one such application, such as Microsoft's NetMeeting and Netscape's CoolTalk.

NetMeeting and CoolTalk are both real-time desktop audio conferencing and data collaboration software applications specifically designed to use the Internet as their communications medium. Both software applications allow a "local" user to place a "call" to a "remote" user located anywhere in the world. With both NetMeeting and CoolTalk, the software application is hosted on a personal computer system at the user's location and on a personal computer system at the remote user's location. Both NetMeeting and CoolTalk require a SLIP (Serial Line Internet Protocol) or PPP (Point-to-Point Protocol) account where Internet access is via a dial-up modem, and where the user, as is typical, accesses the Internet through an ISP (Internet Service Provider). Both NetMeeting and CoolTalk require personal computer systems for the resources necessary to run these applications (e.g., processing power, memory, communications hardware, and the like). In addition, both NetMeeting and CoolTalk require the one user to input an IP (Internet Protocol) address for the other user in order to establish communication between the users.

To facilitate the process of obtaining appropriate Internet addresses, CoolTalk, for example, allows on-line users to list their respective IP addresses with a proprietary CoolTalk central Web server. This allows a user to obtain a list of users currently on-line to whom communication can be established. Upon locating the desired remote user in the Internet address list maintained by the Web server, the local user places the call.

In this manner, the proprietary CoolTalk Web server maintains a user-viewable and user-updated "address book" in which users list their respective Internet addresses and in which they search for the Internet addresses of others with whom they wish to communicate. However, both NetMeeting and CoolTalk require active user input, in that each require the user to input his current Internet address, and in that each require the local user to search the address book for the Internet address of the remote user to be contacted. This can be quite problematic in the case where users obtain access to the Internet via dial-up connections and hence have different Internet addresses each time their respective dial-up connections are established.

In a manner similar to Internet telephony, Internet desktop video conferencing is another prior art application which uses the Internet as its communications medium. One such application, for example, is CU-SeeMe by White Pine. CU-SeeMe provides real time video conferencing between two or more users. As with NetMeeting and CoolTalk, CU-SeeMe is a software application which runs on both the local user's personal computer system and the remote user's personal computer system. The personal computer systems provide the resources for running the application. As with NetMeeting and CoolTalk, CU-SeeMe requires the local user to enter the IP address of the remote user. Like CoolTalk, CU-SeeMe facilitates this process by allowing on-line users to list their respective IP addresses with a proprietary central Web server such that the addresses can be easily indexed and searched.

Another prior art example of remote access via the Internet is status queries of remote devices using the Internet as the communications medium. A typical prior art applica-

3

tion involves interfacing a remote device with a computer system, and accessing the computer system via the Internet. For example, a vending machine can be remotely accessed to determine its status (e.g., the number of sales made, whether the machine needs refills, whether the machine needs maintenance, and the like). The machine is appropriately equipped with sensors, switches, and the like, which in turn are interfaced to a computer system using a software driver. The computer system is coupled to the Internet and interfaces with the machine through the driver, making the relevant information available over the Internet using Web server software. Hence, any interested user (e.g., the vending machine service company) is able to remotely ascertain the status of the machine via the Internet.

A problem with the above described prior art applications is that access to the Internet and communication thereon require a separate host computer system (e.g., a personal computer system) on each side of the Internet connection in addition to the server computer system on the Internet. The two host computer systems provide the computational resources to host the respective software applications, the Internet access software, and any necessary device drivers. The required computational resources consume a significant amount of memory. Because of this, among other reasons, the above prior art applications are not easily transferred to the realm of easy-to-use, intuitive, consumer electronic devices such as digital cameras, which are small in size and so generally constrained by the amount of memory they can house. In addition, a consumer electronic device such as a digital camera that requires a separate computer system would be more expensive and complex, and therefore would not be consistent with the desire of consumers for lower cost and simpler devices.

Also, separate host computer systems (where the host computer systems host the software and drivers required by prior art applications as described above) require extra effort to administer, particularly with regard to networks consisting of a large number of computer systems (e.g., digital cameras each incorporating a computer system). For example, an upgrade to the software residing on each computer system has to be individually installed on each computer system. Also, each computer system has to be individually polled to query whether the computer system has data of interest to the user, and then the data have to be separately accessed and collected from each computer system, then compiled. For example, in an application involving digital cameras, a user may be interested in finding out which digital cameras have images in storage. In the prior art, the user has to access each digital camera individually. In another case, a user may have an interest in maintaining a record of transactions between all users and all digital cameras. Again, in the prior art this is accomplished by individually accessing each digital camera (or, alternatively, each user's computer system) to collect the data, and then compiling the complete list of transactions.

Another problem with the prior art is the fact that the applications described above require the user to know the Internet address of the person or device that is being contacted. The Internet telephony applications (e.g., CoolTalk) often employ a user-viewable and user-updated address book to facilitate the process of locating and obtaining the correct Internet address; however, they require active user input. This is difficult in the case where users obtain access to the Internet via dial-up connections, and thus have changing Internet addresses. Still another problem with the prior art is that the applications described above provide only a limited degree of functionality; that is, they are

4

limited to either chat, video conferencing, or the like. As such, they are not capable of establishing a connection between any type of user system and remote device.

One prior art system is described in the copending previously filed patent application, assigned to the assignee of the present invention, entitled "A Method and System for Hosting an Internet Web Site on a Digital Camera," Eric C. Anderson and others, Ser. No. 09/044,644. This prior art system presents one solution to the problem of gaining remote access to those digital devices where the location and Internet address of the device are highly changeable. This prior art system incorporates a Web server into the digital device, specifically a digital camera. However, the disadvantage to this prior art system is that the Web server consumes valuable memory and computational resources in the digital camera. In addition, because of the limited memory in a device such as a digital camera, the Web server is not as powerful as a Web server on a server computer system.

Thus, a need exists for an inexpensive and powerful method for implementing remote access to digital devices, such as digital cameras, where the location and Internet address of the device are highly changeable. A further need exists for an intuitive, simple protocol for presenting the device's functionality and capabilities to users. In addition, a need exists for a method of efficiently administering a plurality of separate devices. A need also exists for an efficient process of obtaining the address of the device that is transparent to the viewer. The present invention provides a novel solution to the above needs.

SUMMARY OF THE INVENTION

The present invention provides an inexpensive and powerful method for implementing remote access to digital devices, such as digital cameras, where the location and Internet address of the device are highly changeable. The present invention further provides an intuitive, simple protocol for presenting the device's functionality and capabilities to users. In addition, the present invention provides a method of efficiently administering a plurality of separate devices. The present invention also provides an efficient process of obtaining the address of a device that is transparent to the user.

In one embodiment, the present invention is an executable program for accessing a digital camera via a communication network using a Web server on a server computer system and a Web browser (or a program of similar function) on a client computer system that are communicatively coupled via the Internet. The address of the digital camera is registered in an executable program on the server computer system. The executable program is accessed by the client computer system. The executable program connects the digital camera and the server computer system. The executable program enables the client computer system and the digital camera to communicate using any protocol used by these devices, thus allowing data (e.g., images) acquired by the digital camera to be transferred to the client computer system.

The executable program can be implemented in a variety of forms. For example, the executable program can be a Java script. Alternatively, the executable program can be a cgi-bin (Common Gateway Interface-binaries).

For example, in the case of a digital camera, the executable program directly communicates commands from the client computer system to the digital camera when both the client computer system and the digital camera are on-line at the same time. Alternatively, if the digital camera is not

5

on-line, the commands from the client computer system are first stored in the server computer system, and then later communicated by the executable program to the digital camera after a connection between the server computer system and the digital camera is made. The capability to store and then forward commands and data is not limited to a digital camera application nor is it limited to a particular data storage format. The data storage format can be any format that is understood by both the client computer system and the digital device.

Images and any other data acquired by the digital camera are accessed by the server computer system using the executable program and directly transferred to the client computer system when both the client computer system and the digital camera are on-line at the same time. Alternatively, if the client computer system is not on-line, the data are first stored by the server computer system, and then later communicated to the client computer system after a connection between the server computer system and the client computer system is made.

It should be noted, however, that the present invention can be readily modified to function in other embodiments, such as, for example, hand-held digital devices, lap top personal computers, and the like, which require an efficient process of obtaining the address of a device that is transparent to the user.

These and other objects and advantages of the present invention will become obvious to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments which are illustrated in the various drawing figures.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not by way of limitation, in the figures of the accompanying drawings in which like reference numerals refer to similar elements and in which:

FIG. 1A shows a block diagram of a remote access system via the Internet in accordance with one embodiment of the present invention.

FIG. 1B shows a block diagram of a remote access system via a Local Area Network in accordance with one embodiment of the present invention.

FIG. 1C shows a block diagram of a digital camera coupled to the Internet via an Internet Service Provider.

FIG. 2 shows a general purpose computer system upon which embodiments of the present invention may be practiced.

FIG. 3 shows a block diagram of a digital camera for use in accordance with the present invention.

FIG. 4 shows a block diagram of a computer system of a digital camera in accordance with one preferred embodiment of the present invention.

FIG. 5 shows a memory map of a dynamic random access memory of a digital camera in accordance with one embodiment of the present invention.

FIG. 6 shows a diagram of the connectivity and application software of a digital camera in accordance with one embodiment of the present invention.

FIG. 7 is a flowchart of a process for remotely accessing a digital camera implemented by an executable program in accordance with one embodiment of the present invention.

FIG. 8 is a flowchart of a process employed by a digital camera for remote access in accordance with one embodiment of the present invention.

6

FIG. 9 is a flowchart of a process employed by a client computer system for remote access of a digital camera in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In the following detailed description of the present invention, numerous specific details are set forth in order to enable one of ordinary skill in the art to make and use the invention, and are provided in the context of a patent application and its requirements. Although the present invention will be described in the context of a digital camera, various modifications to the preferred embodiment will be readily apparent to those skilled in the art, and the generic principles herein may be applied to other embodiments. That is, any digital device which displays data, images, icons and/or other items, could incorporate the features described below and that device would be within the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

Some portions of the detailed descriptions which follow are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computer memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is here, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computer system. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, bytes, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that throughout the present invention, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes (e.g., the processes of FIGS. 7, 8 and 9) of a computer system or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present invention provides a method for making a digital device (e.g., a digital camera) and its internally stored data remotely accessible over a communication network such as the Internet or a Local Area Network (LAN). The present invention is an executable program placed on a server computer system (specifically, a Web server) that implements commands initiated by a client (or user) using a client computer system with a Web browser or a program of similar function. The present invention enables the digital camera to be set to continuously take pictures of scenes/

7

items of interest and allow a client to access those pictures at any time. The present invention allows the client to access the digital camera from virtually an unlimited number of locations and with the camera in virtually any location.

Referring now to FIG. 1A, a block diagram of communication network 100 is shown. Communication network 100 provides a method in accordance with one embodiment of the present invention which implements remote access to camera 300 and its internally stored data. Communication network 100 includes camera 300, Internet Service Provider (ISP) 110, Internet Service Provider 115, client (or user) computer system 120, and server computer system 190. ISP 110 and ISP 115 are both directly coupled to the Internet 150. Client computer system 120 includes Web browser 121 or a program of similar function, and server computer system 190 includes Web server 161. Web browser 121 interprets HTML (HyperText Mark-up Language) documents and other data retrieved by Web server 161.

In the present embodiment of the present invention, an executable program resides on server computer system 190, specifically on Web server 161. The executable program implements and manages the connection between server computer system 190, client computer system 120, and camera 300. The executable program can be implemented as a Java servlet, as a cgi-bin (Common Gateway Interface-binaries), or as a similar type of application.

With reference still to FIG. 1A, client computer system 120 is communicatively coupled to ISP 115 via a POTS (plain old telephone system) dial-up connection. Client computer system 120 is coupled to the Internet 150 via one of a bank of modems maintained on the premises of ISP 115. ISP 115 is coupled directly to the Internet via an all-digital connection (e.g., a T1 line). However, other means of coupling client computer system 120 to the Internet 150 may be used in accordance with the present invention.

As depicted in FIG. 1A, camera 300 is communicatively coupled to server computer system 190 via the Internet 150 using a dial-up connection to ISP 110 via a POTS line. Digital camera 300 accesses ISP 110 using a modem, coupling to one of a bank of modems maintained on the premises of ISP 110. ISP 110 is in turn coupled directly to the Internet 150 via an all-digital connection. However, other means of coupling camera 300 to the Internet 150 may be used in accordance with the present invention.

With reference still to FIG. 1A, it should be further appreciated that while communication network 100 shows camera 300 coupling to Internet 150 via one ISP (e.g., ISP 110) and user 120 coupling to Internet 150 via a separate ISP (e.g., ISP 115), user 120 and camera 300 could be coupled to Internet 150 through a single ISP. In such a case, user 120 and camera 300 would be coupled to two separate access ports (e.g., two separate modems out of a bank of modems) of the same ISP.

With reference now to FIG. 1B, in another embodiment of the present invention, the communication network is comprised of Local Area Network (LAN) 170. For example, LAN 170 may be a communication network located within a firewall of an organization or corporation. Client (or user) computer system 171 and server computer system 172 are communicatively coupled via communication line 173a. Client computer system 171 includes an application that is analogous to a Web browser for interpreting HTML documents and other data. Similarly, server computer system 172 includes an application analogous to a Web server for retrieving HTML documents and other data. Camera 300 can be coupled to server computer system 172 through any

8

of a variety of means known in the art. For example, camera 300 can be coupled to server computer system 172 via communication line 173b of LAN 170. This coupling can be accomplished over any network protocol that supports a persistent network connection, such as TCP (Transmission Control Protocol), NetBIOS, IPX (Internet Packet Exchange), and LU6.2, and link layers protocols such as Ethernet, token ring, and ATM (Asynchronous Transfer Mode). Alternatively, camera 300 can be coupled to server computer system 172 via an input/output port (e.g., a serial port) of server computer system 172.

Referring now to FIG. 1C, a more detailed diagram of camera 300 coupled to the Internet 150 is shown. Camera 300 is coupled to an external modem 101. Camera 300 is coupled to modem 101 via any of several communications means (e.g., Universal Serial Bus, infrared link, and the like). Modem 101 is in turn coupled to a POTS telephone jack 102 at the camera's location. Telephone jack 102 couples modem 101 to one of modems 103 of ISP 110 via the telephone company's local loop. ISP 110 is directly coupled to the Internet 150 via an all digital connection (e.g., a T1 line).

Continuing with reference to FIG. 1C, modem 101 is shown as an external modem. However, the functionality of modem 101 can be implemented directly within the electronics of camera 300 (e.g., via a modem application-specific integrated circuit or ASIC), or alternatively can be implemented as a software-only modem executing on a computer within camera 300. As such, it should be appreciated that, at the hardware connectivity level, modem 101 can take several forms. For example, a wireless modem can be used in which case the camera is not connected via an external wire to any land line. Alternatively, there may be applications in which camera 300 includes suitable electronic components enabling a connection to a conventional computer system network (e.g., Ethernet, AppleTalk, and the like), which is in turn directly connected to the Internet (e.g., via a gateway, a firewall, and the like), thereby doing away with the requirement for an ISP. Hence, it should be appreciated that the present invention is not limited to any particular method of accessing the Internet 150.

Refer now to FIG. 2 which illustrates server computer system 190 upon which embodiments of the present invention may be practiced (the following discussion is also pertinent to a client computer system). In general, server computer system 190 comprises bus 200 for communicating information, processor 201 coupled with bus 200 for processing information and instructions, random access memory 202 coupled with bus 200 for storing information and instructions for processor 201, read-only memory 203 coupled with bus 200 for storing static information and instructions for processor 201, data storage device 204 such as a magnetic or optical disk and disk drive coupled with bus 200 for storing information and instructions, optional display device 205 coupled to bus 200 for displaying information to the computer user, optional alphanumeric input device 206 including alphanumeric and function keys coupled to bus 200 for communicating information and command selections to processor 201, optional cursor control device 207 coupled to bus 200 for communicating user input information and command selections to processor 201, and network interface card (NIC) 208 coupled to bus 200 for communicating from a communication network to processor 201.

Display device 205 of FIG. 2 utilized with server computer system 190 of the present invention may be a liquid crystal device, cathode ray tube, or other display device

suitable for creating graphic images and alphanumeric characters recognizable to the user. Cursor control device 207 allows the computer user to dynamically signal the two-dimensional movement of a visible symbol (pointer) on a display screen of display device 205. Many implementations of the cursor control device are known in the art including a trackball, mouse, joystick or special keys on alphanumeric input device 206 capable of signaling movement of a given direction or manner of displacement. It is to be appreciated that the cursor control 207 also may be directed and/or activated via input from the keyboard using special keys and key sequence commands. Alternatively, the cursor may be directed and/or activated via input from a number of specially adapted cursor directing devices.

Referring now to FIG. 3, a block diagram of digital camera 300 is shown for use in accordance with the present invention. Camera 300 preferably comprises imaging device 314, system bus 316 and computer 318. Imaging device 314 is optically coupled to object 312 and electrically coupled via system bus 316 to computer 318. Once a photographer has focused imaging device 314 on object 312 and, using a capture button or some other means, instructed camera 300 to capture an image of object 312, computer 318 commands imaging device 314 via system bus 316 to capture raw data representing object 312. The captured raw data are transferred over system bus 316 to computer 318, which performs various processing functions on the data before storing it in memory. System bus 316 also passes various status and control signals between imaging device 314 and computer 318.

Referring now to FIG. 4, a block diagram of one embodiment of computer 318 is shown. System bus 316 provides connection paths between imaging device 314, an optional power manager 442, central processing unit (CPU) 444, dynamic random-access memory (DRAM) 446, input/output interface (I/O) 448, non-volatile memory 450, and buffers/connectors 452. Removable memory 454 connects to system bus 316 via buffers/connectors 452. Alternatively, camera 300 may be implemented without removable memory 454 or buffers/connectors 452.

Power manager 442 communicates via line 466 with power supply 456 and coordinates power management operations for camera 300. CPU 444 typically includes a conventional processor device for controlling the operation of camera 300. In the present embodiment, CPU 444 is capable of concurrently running multiple software routines to control the various processes of camera 300 within a multi-threaded environment. DRAM 446 is a contiguous block of dynamic memory which may be selectively allocated to various storage functions. LCD controller 490 accesses DRAM 446 and transfers processed image data to LCD screen 402 for display.

I/O 448 is an interface device allowing communications to and from computer 318. I/O 448 permits an external device (not shown) to connect to and communicate with computer 318. I/O 448 also interfaces with a plurality of buttons and/or dials 404, and optional status LCD 406, which in addition to LCD screen 402, are the hardware elements of the camera's user interface 408.

Non-volatile memory 450, which may typically comprise a conventional read-only memory or flash memory, stores a set of computer-readable program instructions to control the operation of camera 300.

Referring now to FIG. 5, a memory map showing one embodiment of dynamic random access memory (DRAM) 446 is shown. In the present embodiment, DRAM 446 includes RAM disk 532, system area 534, and working memory 530.

RAM disk 532 is a memory area used for storing raw and compressed data and typically is organized in a "sectored" format similar to that of conventional hard disk drives. In the present embodiment, RAM disk 532 uses a well-known and standardized file system to permit external devices, via I/O 448 of FIG. 4, to readily recognize and access the data stored on RAM disk 532. System area 534 typically stores data regarding system errors (for example, why a system shutdown occurred) for use by CPU 444 (FIG. 4) upon a restart of computer 318 (FIG. 3).

Working memory 530 includes various stacks, data structures and variables used by CPU 444 while executing the software routines used within computer 318. Working memory 530 also includes several input buffers 538 for temporarily storing sets of raw data received from imaging device 314 (FIG. 3), and frame buffer 536 for storing data for display on LCD screen 402 (FIG. 4). In the present embodiment, each input buffer 538 and frame buffer 536 are split into two separate buffers (shown by the dashed lines) to improve the display speed of the digital camera and to prevent the tearing of the image in LCD screen 402.

Referring now to FIG. 6, a diagram of the connectivity and application software of camera 300 of FIG. 3 is shown. At the software level, computer 318 (FIG. 3) of camera 300 hosts any network protocol that supports a persistent network connection. This coupling can be accomplished over any network protocol that supports a persistent network connection, such as TCP/IP (Transmission Control Protocol/Internet Protocol) including Point-to-Point Protocol, NetBIOS, IPX, and LU6.2, and link layers protocols such as Ethernet, token ring, and ATM. Protocol stack 601 interfaces with the communications hardware 602 (e.g., a modem) of camera 300 and the application layer 603. The bottom of protocol stack 601 includes communication hardware interface drivers which interface directly with the various communications hardware with which camera 300 must function (e.g., a Universal Serial Bus and the like). Applications layer 603, protocol stack 601, and operating system 604 are installed as software modules in DRAM 446 (FIG. 4) of camera 300. Software applications within applications layer 603 interface with operating system 604. Operating system 604 controls the hardware functionality of camera 300 (e.g., taking pictures, storing pictures, and the like) via camera hardware 605. Incoming data 620, such as HTTP (Hypertext Transfer Protocol) requests and the like, are received and outgoing data 630, such as HTML (Hypertext Markup Language) files and the like, are transferred to and from camera 300 via protocol stack 601 and communications hardware 602. Web browser 121 of FIG. 1A (or a program of similar function) can process data files, launch plug-ins, and run Java applets that communicate with camera 300 in a variety of methods in addition to those methods involving an exchange of HTML files.

FIG. 7 illustrates an executable program 700 for client computer system 120 (specifically, Web browser 121 of FIG. 1A or a program of similar function) to remotely access camera 300 (FIG. 3), where executable program 700 is implemented in accordance with the present invention as program instructions stored in computer-readable memory units (e.g., read-only memory 203) and implemented by processor 201 of server computer system 190 of FIG. 1A (specifically, by Web server 161). Executable program 700 performs functions both for and in response to Web browser 121 (or a program of similar function) and camera 300. The description below first discusses the steps associated with setting up a connection between executable program 700 and camera 300, then discusses the steps associated with

11

setting up a connection between executable program 700 and Web browser 121 (or a program of similar function), however, the present invention is not limited by the order in which these steps are presented.

In the present embodiment, executable program 700 is identified and accessed by its own unique address, commonly referred to as an URL (Unified Resource Locator), as is well known in the art. The URL for executable program 700 fully describes where it resides on a communication network (e.g., the Internet 150) and how it is accessed. In the present embodiment, included in the URL for executable program 700 is the name of camera 300. Accordingly, in one embodiment using a servlet for executable program 700, a standard format for a URL is: `http://webserver/HostName/cameraServlet/WellKnownName/cameraName`.

In step 705, executable program 700 receives and accepts a connection request from camera 300. Executable program 700 runs constantly on Web server 161 and is configured to listen for connection requests on a plurality of communication protocols (e.g., TCP, NetBIOS, and the like). In the present embodiment, camera 300 is connected to executable program 700 via Web server 161 as described in conjunction with FIG. 1A, and opens the connection using the connectivity and application software described in conjunction with FIG. 6.

In step 710, executable program 700 receives and reads registration information from camera 300. As mentioned above, executable program 700 is configured to communicate using a number of different communication protocols. Such registration information includes the name of the camera and authentication information such as security information and account information. Executable program 700 uses the camera name to identify the camera and locate it in response to a client request.

In step 720, executable program 700 compares the registration information with a predefined access control list to determine if camera 300 is a camera for which Web server 161 is to provide support and service. If not, executable program 700 closes the connection between Web server 161 and camera 300.

In step 725, upon successful completion of step 720, executable program 700 registers camera 300 and stores the camera's name and associated requirements, such as security and account information. Executable program 700 also sends a message that camera 300 is registered. At this point, the connection between executable program 700 can be either terminated or maintained at the option of the camera's operator. If the connection is terminated, the registration information is maintained by Web server 161 and can be later accessed by executable program 700 when a subsequent connection is made with camera 300.

In step 730, executable program 700 accepts the connection request from camera 300 and thus has a persistent and long term connection with camera 300. As described above, the connection can be an ongoing connection maintained from the time when camera 300 was first registered. New connections can be made in the future whenever camera 300 reinitiates the registration protocol. Once camera 300 and executable program 700 have established a connection, they then wait until a client also makes a connection to access the camera. However, as will be described below in conjunction with step 770, it is not necessary for there to be an open connection between the Web server and the camera at the same time that there is an open connection between the Web browser (or a program of similar function) and Web server to accomplish remote access of the camera in accordance with the present invention.

12

In step 740, executable program 700 receives and accepts a request for a connection from a client. The client enters the URL of executable program 700, including the name of the camera (e.g., camera 300) to which access is desired, into a Web browser (e.g., Web browser 121 on client computer system 120 of FIG. 1A or a program of similar function). Using standard communication protocols such as TCP/IP, Web server 161 is queried with the URL for executable program 700. Web server 161 recognizes the URL and makes the connection to executable program 700. Thus, the present invention establishes a single location identified by a known URL where the client always goes to connect to the camera, no matter where the camera is or where the client is. (For those cases in which TCP/IP is not available—for example, when the device is not attached to the Internet or the like—camera 300 connects to executable program 700 directly using a well-known address that is associated with the protocol being used, such as a NetBIOS name.)

In step 745, executable program 700 then assumes control over the TCP/IP connection between Web browser 121 (or a program of similar function) and Web server 161. Executable program 700 establishes a persistent and long term connection between the browser and server. That is, the connection between Web browser 121 (or a program of similar function) and Web server 161 is kept open by executable program 700. As will be described below in conjunction with step 770, it is not necessary for there to be an open connection between the client computer system and the Web server at the same time that there is an open connection between the Web server and the camera to accomplish remote access of the camera in accordance with the present invention.

In step 750, executable program 700 next determines if camera 300 is registered as discussed in conjunction with step 735. If camera 300 is not registered, executable program 700 sends an appropriate message to the client to indicate such.

In step 760, if the camera is registered, executable program 700 validates the required access information provided by the client against the security and account information provided when camera 300 was registered. For example, executable program 700 validates whether the client is utilizing an authorized password or user name. If not, executable program 700 transmits an appropriate message to the client. The present invention can optionally provide additional services related to security or account information. For example, it could control the type of access a client is permitted based on the authentication information received from the client, or it could verify credit information and bill the client for services requiring payment.

In step 765, upon satisfactory completion of step 760, executable program 700 extends the connection from Web browser 121 (or a program of similar function) to camera 300 if there is an established connection to the camera as described in conjunction with step 730. Hence, a client using Web browser 121 or a program of similar function has direct, remote access to camera 300 via executable program 700 in Web server 161.

In step 770, executable program 700 forwards commands from a client to camera 300, and forwards images and other data from camera 300 to the client via Web server 161 and Web browser 121 (or a program of similar function). That is, executable program 700 enables a direct communication between the client computer system and the camera allowing the client to remotely access and manage the camera. If the client and the camera are both concurrently connected to

13

executable program 700, then the client immediately receives the data, and camera 300 immediately executes any commands from the client.

However, if Web browser 121 (or a program of similar function) and camera 300 are not each connected at the same time, remote access to the camera is still accomplished in accordance with the present invention. If camera 300 is not on-line, a client uses Web browser 121 (or a program of similar function) to access Web server 161 and executable program 700. The client transmits commands, and executable program 700 stores the commands on Web server 161. The client may then close the connection to the Web server. Subsequently, when camera 300 opens a connection to Web server 161 and executable program 700, executable program 700 retrieves the commands and forwards them to the camera. Camera 300 downloads the commands and executes them. The results from executing the commands are then sent to executable program 700, which stores them on Web server 161 until they are retrieved by the client.

Similarly, if camera 300 establishes a connection with Web server 161 and executable program 700 but the client is not on-line, the camera can, for example, download images and any other data that executable program 700 stores on Web server 161. Camera 300 may then close the connection to the Web server. The client then later makes a connection to Web server 161 and executable program 700, which retrieves the data and forwards it to the client. The client can also enter commands at this time, which are stored by executable program 700 as described above.

In summary, with references to FIG. 1A and FIG. 7, the client, via Web browser 121 or a program of similar function, Web server 161 and executable program 700, accesses camera 300 to request and retrieve data. Web browser 121 or a program of similar function inserts the Internet address inside the data request (e.g., a HTTP request) and sends the request to Web server 161. Web server 161 receives the data request and associates the request with executable program 700, which in turn assumes the connection between Web server 161 and Web browser 121 (or a program of similar function), and which also establishes a connection between Web browser 121 (or a program of similar function) and camera 300. Executable program 700 subsequently forwards commands from the client to camera 300, and retrieves the requested data (e.g., a HTML file) containing the data and sends it back to Web browser 121 (or a program of similar function). Web browser 121 (or a program of similar function) then interprets the commands and displays the resulting image. The process of accessing a data file from a Web server is commonly referred to as accessing a Web page. Similarly, the process of sending data files from a Web server to a Web browser is commonly referred to as sending a Web page.

Thus, as described above, the present invention provides an intuitive and easy-to-use interface enabling remote access between a client and a camera. By functioning with widely used and familiar Web browsers (or programs of similar function) using standard format URLs to identify the executable program and camera, the present invention provides a simple and familiar interface for accessing the camera. By registering the camera and using an unchanging URL, name to identify the executable program and the camera, the present invention enables the client to locate and access the camera from any remote location no matter where the camera is located. In addition, by using a Java servlet or a cgi-bin for the executable program, the present invention is supported by commonly used Web servers and is readily implemented.

14

Executable program 700 is located on the Web server and not on camera 300, so it does not require additional and substantial memory dedicated to enabling remote access. As such, the present invention permits remote access within the constraints of the size of the camera. In addition, in accordance with the present invention, camera 300 does not require a separate, external computer system (e.g., a personal computer system) for connecting to ISP 110 (FIG. 1A) or for implementing commands and transmitting data, thus providing an inexpensive method for providing remote access to cameras.

Also, by locating the present invention on a Web server (e.g., Web server 161 of FIG. 1A), the Web server becomes a focal point for accessing and managing a plurality of cameras that otherwise would have to be managed and configured separately. For example, executable program 700 on Web server 161 could be updated with new software, and in effect all cameras accessed through that executable program would automatically be updated as well. As another example, executable program 700 could be configured to compile data regarding interactions between clients and all cameras accessed by the executable program. In another example, in accordance with the present invention, a client needs to go only to a single location to determine which of a plurality of cameras served by the executable program have data that have been downloaded to the Web server. Thus, instead of having to access a number of cameras separately, the present invention establishes a single location from which a client can access information about several cameras.

By functioning with a Web-based interface and widely used Web browsers (or programs of similar function), the present invention provides a simple, intuitive, and familiar interface for accessing camera 300's functionality. Accordingly, camera 300's controls and functions are intuitively easy to utilize. Since Web pages and their associated controls (e.g., push buttons, data entry fields, and the like) are very familiar to most users, the remote access functionality of camera 300 can be utilized without requiring a extensive learning period for new users. For example, a consumer purchasing a remotely accessible camera is typically able to easily and immediately use the remote accessibility functions with minimal set-up.

FIG. 8 illustrates a process 800 for remotely accessing camera 300 (FIG. 3), where process 800 is implemented as program instructions stored in computer-readable memory units (e.g., non-volatile memory 450 of FIG. 4) and implemented by CPU 444 (FIG. 4) of camera 300 in accordance with the present invention. FIG. 8 illustrates the process for remotely accessing a camera from the perspective of the actions performed by the camera and the camera operator in accordance with the present invention.

In step 805, camera 300 of FIG. 3 opens a connection to communication network 100 of FIG. 1A. In the present embodiment, camera 300 is coupled to server computer system 190 (specifically, Web server 161) as described in conjunction with FIG. 1A, and opens the connection using the connectivity and application software described in conjunction with FIG. 6. As described in conjunction with FIG. 1C, in the present embodiment, camera 300 couples directly to the telephone system such that a separate and dedicated computer system (e.g., a personal computer system) is not necessary.

In step 810, with the connection made to Web server 161, camera 300 requests a connection to executable program 700 (FIG. 7). For cases in which camera 300 is accessing

15

executable program 700 via, for example, a TCP/IP network such as the Internet 150, then executable program 700 is identified with a URL that is used by the camera to access the executable program. For those cases in which TCP/IP is not available (e.g., when the device is not attached to the Internet or the like), camera 300 connects to executable program 700 directly using a well-known address that is associated with the protocol being used, such as a NetBIOS name.

In step 820, with the camera connected to the executable program, camera 300 registers with executable program 700. For example, camera 300 provides information including an identification name and authentication information such as a password and account information. The information is electronically transmitted from camera 300 and read by executable program 700. Based on this information, the connection between camera 300 and executable program 700 is established if the camera is of the type that is designated to be supported by Web server 161.

In step 830, camera 300 and executable program 700 are linked via a persistent and long term connection; that is, the connection remains open awaiting a client to request access to the camera via Web server 161. As discussed above in conjunction with FIG. 7, it is not necessary for the camera and a client to be connected at the same time to executable program 700.

In step 835, camera 300 receives commands from and transmits data to a client using a Web browser or a program of similar function on a client computer system (e.g., client computer system 120 and Web browser 121 of FIG. 1A or a program of similar function). As described above in conjunction with FIG. 7, the commands and data can be transmitted through an active connection or stored on the Web server.

In the present embodiment, camera 300 is provided with several different operating modes for supporting various camera functions. In capture mode, camera 300 supports the actions of preparing to capture an image and of capturing an image. In review mode, camera 300 supports the actions of reviewing camera contents, editing and sorting images, and printing and transferring images. In play mode, camera 300 allows the client to view screen-sized images in the orientation that the image was captured. Play mode also allows the client to hear recorded sound associated to a displayed image, and to play back sequential groupings of images, which may comprise time lapse, slide show, and burst image images. The client preferably switches between the capture, review, and play modes.

Camera 300 is capable of implementing a wide variety of remote access and remote imaging/surveillance applications. In the present embodiment, camera 300 only records successive images for remote access by the client. The images are loaded into the camera's memory on a first-in, first-out (FIFO) basis, with the earliest recorded image being replaced by the latest recorded image. The number of images available to the client depends upon the amount of installed memory in the camera.

With reference still to step 835 of FIG. 8, when the client and the camera are connected to the Web server at the same time, the commands can be transmitted directly from Web browser 121 (or a program of similar function) to camera 300 via executable program 700 on Web server 161, and camera 300 then executes the commands on-line. Alternatively, when camera 300 is not connected to executable program 700, a client can store commands on Web server 161 by accessing the Web server in a normal fashion,

16

then entering the commands to be stored via executable program 700. Camera 300 then executes the commands when it subsequently connects with executable program 700.

Also in step 835, camera 300 transmits data to a client. Similar to the above, data from camera 300 can be transmitted directly to Web browser 121 or a program of similar function via executable program 700 on Web server 161, when the client and the camera are connected to the Web server at the same time. Alternatively, when a client is not connected to executable program 700, the data are stored on Web server 161 by executable program 700, which then retrieves and transmits the data to a client when the client subsequently connects with executable program 700.

This process of accessing camera 300 from Web browser 121 or a program of similar function occurs transparently with respect to the client. In a typical case, for example, the client types the URL for executable program 700 (which includes the name of camera 300) into Web browser 121 (or a program of similar function) and his "enter" or "return". In accordance with the present invention, the next Web page the client views is the image generated by the data returned from camera 300. Beyond entering the URL for executable program 700 including camera 300, no further action from the client is required in order to access the Web pages hosted by camera 300.

FIG. 9 illustrates a process 900 for remotely accessing camera 300 (FIG. 3), where process 900 is implemented as program instructions stored in computer-readable memory units (e.g., read-only memory) and implemented by the central processor of client computer system 120 (specifically, Web browser 121 or a program of similar function) of FIG. 1A. FIG. 9 illustrates the process for remotely accessing a camera from the perspective of the actions performed by the client in accordance with the present invention.

In step 905, the client opens a network connection such as a dial-up connection to ISP 115 of FIG. 1A.

In step 910, the client enters into a Web browser (e.g., Web browser 121 on client computer system 120 of FIG. 1A or a program of similar function) the URL of executable program 700, including the name of the camera (e.g., camera 300) to which access is desired. Alternatively, the client enters the URL of executable program 700 only. (For those cases in which TCP/IP is not available—for example, when the device is not attached to the Internet or the like—camera 300 connects to executable program 700 directly using a well-known address that is associated with the protocol being used, such as a NetBIOS name.) Using standard communication protocols such as TCP/IP, Web server 161 is queried with the URL for executable program 700. Web server 161 recognizes the URL and makes the connection to executable program 700. A persistent and long term two-way connection is opened between the client on Web browser 121 (or a program of similar function) and executable program 700.

In step 915, the client enters the authentication information required to gain access to executable program 700 and camera 300.

In step 920, as explained above, from his/her remote location the client sends commands to camera 300 and receives images from the camera. As explained above, the client and the camera do not have to be connected to the Web server at the same time to enable remote access. Depending on the particular application, the Web page for camera 300 can include control buttons, data entry fields, drop-down menus, and the like.

17

Thus, executable program 700 enables the client to access from a remote location the functional controls of camera 300 as well as the images and other data acquired by the camera.

Pseudo-Code Sections A, B, C, D and E below provide additional details regarding processes 700, 800 and 900 of FIGS. 7, 8 and 9. Pseudo-Codes Sections A through E represent the method of one embodiment of the present invention. However, it is appreciated that other embodiments are possible in accordance with the present invention.

18

With references to FIGS. 7 and 9, the pseudo-code for the connection of a client to the server computer system, specifically to executable program 700, is described in accordance with one embodiment of the present invention in Pseudo-Code Section A below.

Pseudo-Code Section A: Client Connection Setup

```

1. Client sends HTTP Post command to http://serverName/gateway device="deviceName"
   Note: authentication/security data is optional and is handled by the browser.
2. Client receives HTTP response from Gateway
3. If response is OK or OKDataPresent
{
    Client has TCP connection it can use to send/receive anything to/from the device
    The protocol used can be any that the Client & Device agree upon.
}
else
{
    The response is failed or DataPresent.
    The TCP connection is closed:
}

```

Note: the OKDataPresent and DataPresent responses are informational. The Client at any time on separate connections may issue a CacheData or GetCache command to the Gateway. See algorithm C.

30 With references to FIGS. 7 and 8, the pseudo-code for the connection of a device (e.g., digital camera 300 of FIG. 1A) to the server computer system, specifically to executable program 700, is described in accordance with one embodiment of the present invention in Pseudo-Code Section B below.

Pseudo-Code Section B: Device Connection Setup/Registration

```

1. Device establishes a connection to the Gateway using any networking protocol
   supported by the Gateway (TCP, NetBIOS, IPX, 802.2, etc) using a known address.
2. Device sends a Register request to the Gateway on the connection passing
   its name (and optionally authentication/security information).
3. Device receives response from Gateway
4. If the response is OK or OKDataPresent
{
    The Device may use the existing connection to wait for requests from clients
    or the Device may optionally close the connection.
}
else
{
    The response is Failed or DataPresent
    The connection is closed.
}

```

Note: the OKDataPresent and DataPresent responses are informational. The Device at any time on separate connections may issue a CacheData or GetCache command to the Gateway. See algorithm D.

60

With reference to FIG. 7, the pseudo-code for executable program 700 in response to a client connection request is described in accordance with one embodiment of the present invention in Pseudo-Code Section C below.

65

Pseudo-Code Section C: Gateway Handles Client Connection Request

```

Start:
    Wait for incoming requests from clients
    if its an HTTP Post command with parameters CacheData and deviceName
    {
        go to CacheData
    }
    else if its an HTTP Get command with parameters GetCache and deviceName
    {
        go to GetCache
    }
    if its an HTTP Post command from a client with the name of device
    {
        go to Connect
    }
    else
    {
        go to Fail
    }
Connect:
    Look up device by name in registry
    if the device entry is found
    {
        if there is a device connection id in the entry
        {
            if the entry is not busy
                go to Success
            else go to Fail
        }
        else
        {
            Attempt to establish connection with device at last known address
            if connection established
            {
                go to Success
            }
            else go to Fail
        }
    }
    else
    {
        send Fail response
        close connection
        go to Start
    }
Fail:
    if there is data in the cache
    {
        set response to DataPresent
    }
    else
    {
        set response to Fail
    }
    send response
    close connection
    go to Start
Success:
    if there is data in the cache
    {
        set response to OkDataPresent
    }
    else
    {
        set response to Ok
    }
    store connection id of the client in the registry entry
    go to Start
CacheData:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        store data in Cache for device with identification information

```

-continued

```

    send Ok response
    close connection
    go to Start;
}
GetCache:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        send data in Cache for this client
        close connection
        go to Start
    }
}

```

With reference to FIG. 7, the pseudo-code for executable program 700 in response to a device connection request is described in accordance with one embodiment of the present invention in Pseudo-Code Section D below.

Pseudo-Code Section D: Gateway Handles Device Requests

```

Start:
    Wait for incoming requests from devices
    Receive incoming request
    if its a Register request
    {
        go to Register
    }
    else if its a CacheData request
    {
        go to CacheData
    }
    else if its a GetCache request:
    {
        go to GetCache
    }
    else
    {
        return Failed
        close connection
        go to Start
    }
Register:
    Note: The gateway may optionally authenticate device
    Get device name from Register request
    look up name in table
    if name is found
    {
        go to Success
    }
    else
    {
        The gateway may optionally add the name or reject the
        registration attempt, then go to Success or Fail
        respectively
    }
Success:
    check for the presence of cached client data
    if cached data is present
    {
        set response to OkDataPresent;
    }
    else
    {
        set response to Ok
    }
}

```

-continued

```

    Store a connection id for the incoming connection in
    the registry entry for the device.
    send response
    go to Start
Fail:
    if cached data from clients is present for this device
    {
        set response to DataPresent
    }
    else
    {
        set response to Fail
    }
    send response
    close connection
    go to Start
CacheData:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        store data in Cache for client
        send Ok response
        close connection
        go to Start;
    }
GetCache:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        send data in Cache for device to the device
        close connection
        go to Start
    }
}

```

With reference to FIG. 7, the pseudo-code for executable program 700, specifically the handling of data when an open connection is present between the device and the client

computer system, is described in accordance with one embodiment of the present invention in Pseudo-Code Section E below.

Pseudo-Code Section E: Gateway Handles Data on Existing Connections

```

Start:
  Wait for data
  Map the incoming connection id to its partner connection id
  if there is incoming data
  {
    Send the data out on the partner connection id
    go to Start
  }
  else if the connection closed
  {
    if it is a client connection
    {
      Remove the client's connection id from the
      device's entry in the registry.
    }
    else
    {
      Remove both the client and device connection ids
      from the device's entry in the registry.
      Close the client's connection
      go to Start
    }
  }
  go to Start
  
```

As described above, the remote accessibility of camera **300** provides for many new applications of digital imagery. One such application involves setting up camera **300** at some remote location and using it to take pictures at successive intervals. These pictures would be accessed via the Internet **150** as they are taken. The interval can be adjusted (e.g., more or less pictures per minute) in response to commands entered by a client via a Web browser (e.g., Web browser **121** of FIG. 1A or a program of similar function).

Another application involves using camera **300** in conjunction with a motion detector. When used in conjunction with a motion detector, camera **300** can be configured to capture an image in response to receiving a signal from the motion detector (e.g., detecting the motion of an intruder), thereby taking a picture of whatever triggered the detector's signal output. Alternatively, camera **300** can detect motion by simply comparing successive images to detect changes between them, thereby dispensing with the need for a separate motion detector.

Yet another application involves using camera **300** in conjunction with a remote aiming device. Camera **300** can be mounted on a remotely operated aiming device (e.g., a motorized tripod). The aiming device is controlled via the Internet **150** in the same manner the camera is controlled via the Internet **150**. Alternatively, camera **300** could be coupled to control the remote aiming device directly. The remote aiming device allows a client to control the field of view of the camera **300** in the same manner the client controls other functionality (e.g., picture resolution, picture interval, and the like).

In this manner, executable program **700** of the present invention is able to implement sophisticated remote surveillance of the type previously performed by expensive, prior art closed circuit television devices. Unlike the prior art, however, executable program **700** is inexpensive and relatively simple to implement.

Thus, the present invention provides a method for making a digital camera and its internally stored data remotely

accessible. The present invention enables the digital camera to be set to continuously take pictures of scenes and items of interest and to allow a user to access those pictures at any time. The present invention implements remote accessibility via a communication network such as the Internet, thus allowing the user to access the digital camera from virtually an unlimited number of locations and with the camera in virtually any location.

A digital camera in accordance with the present invention does not require a separate, external computer system (e.g., a personal computer system) for Internet connectivity, thus providing an inexpensive method for making remotely accessible digital cameras widely available. In addition, a digital camera in accordance with the present invention is accessed via the widely used and very familiar Web browser (or a program of similar function). By functioning with typical, widely used Web browsers (or programs of similar function), the present invention provides a simple, intuitive, and familiar interface for remotely accessing the digital camera's functionality and capabilities. In so doing, the controls and functions of the digital camera are intuitively easy to utilize, and do not require an extensive learning period for new users. The present invention also provides an efficient and user-transparent process of obtaining the address of a digital camera. Also, the present invention provides a method for efficiently administering a plurality of separate digital cameras.

Although the present invention is described in the context of a digital camera, it is not limited to this embodiment. Hence, the present invention does not provide only a limited degree of functionality as in the prior art applications; that is, it is not limited to either chat or video conferencing, or the like. As such, the present invention is capable of establishing a connection between any type of client system and remote device.

The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.

What is claimed is:

1. A method for allowing a client computer to remotely access a digital image capture unit via a communication network wherein a server computer system and said client computer system are communicatively coupled with communication equipment, said method comprising the steps of:
 - a) allowing said digital image capture unit to establish communication with said server computer system over said network, wherein said digital image capture unit includes connectivity software that enables said digital image capture unit to establish a network connection with said server computer system;
 - b) receiving an address of said digital image capture unit and registering said address in an executable program on said server computer system, wherein said digital image capture unit automatically registers said address with said server computer system;
 - c) allowing said client computer system to access said executable program on said server computer system; and

25

- d) establishing direct communication between said client computer system and said digital image capture unit by communicating commands between said client computer system and said digital image capture unit via said executable program, such that data captured by said digital image capture unit is transferred to said client computer system via said server computer system.
2. The method of claim 1 wherein said communication network is the Internet.
3. The method of claim 1 wherein said communication network is a Local Area Network.
4. The method of claim 1 wherein said image capture unit is a digital camera.
5. The method of claim 1 wherein step a) further comprises communicating authentication information between said digital image capture unit and said executable program.
6. The method of claim 1 wherein said executable program is a Java servlet.
7. The method of claim 1 wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).
8. The method of claim 1 wherein step c) further comprises connecting said digital image capture unit to said server computer system via a Local Area Network.
9. The method of claim 1 wherein step c) further comprises connecting said digital image capture unit to said server computer system via an input/output port of said server computer system.
10. The method of claim 1 wherein step c) further comprises connecting said digital image capture unit to said server computer system via the Internet.
11. The method of claim 1 wherein step d) further comprises storing said commands in a memory unit of said server computer system and communicating said commands to said digital image capture unit at a time when a connection is made between said server computer system and said digital image capture unit.
12. The method of claim 1 further comprising the steps of:
- e) accessing via said executable program data acquired by said digital image capture unit; and
 - f) transferring said data from said digital image capture unit to said client computer system via said server computer system.
13. The method of claim 12 further comprising storing said data in a memory unit of said server computer system and communicating said data to said client computer system at a time when a connection is made between said server computer system and said client computer system.
14. A computer system comprising:
- a) a processor coupled to a bus; and
 - a memory unit coupled to said bus and having stored therein an executable program that when executed by said processor implements a method for allowing a client computer to remotely access a digital image capture unit via a communication network, said method comprising the steps of:
 - a) allowing said digital image capture unit to establish communication with said server computer system over said network, wherein said digital image capture unit includes connectivity software that enables said digital image capture unit to establish a network connection with said server computer system;
 - b) receiving an address of said digital image capture unit and registering said address with said executable program, wherein said digital image capture unit automatically registers said address with said server computer system;

26

- c) allowing said client computer system to access said executable program over said communication network; and
 - d) establishing direct communication between said client computer system and said digital image capture unit by communicating commands between said client computer system and said digital capture unit via said executable program.
15. The computer system of claim 14 wherein said computer system is a server computer system.
16. The computer system of claim 14 wherein said digital image capture unit is a digital camera.
17. The computer system of claim 14 wherein said executable program is a Java servlet.
18. The computer system of claim 14 wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).
19. The computer system of claim 14 wherein said digital image capture unit and said computer system are connected via a Local Area Network.
20. The computer system of claim 14 wherein said digital image capture unit and said computer system are connected via an input/output port of said computer system.
21. The computer system of claim 14 wherein said digital image capture unit and said computer system are connected via the Internet.
22. In a communication network comprising a server computer system and a client computer system communicatively coupled with communication equipment, a method for allowing said client computer to remotely access a digital camera via said communication network, said method comprising the steps of:
- a) allowing said digital camera to establish communication with said server computer system over said communication network, wherein said digital camera includes connectivity software that enables said digital camera to establish a network connection with said server computer system;
 - b) receiving an address of said digital camera and registering said address in an executable program on said server computer system, wherein said digital camera automatically registers said address with said server computer system;
 - c) allowing said client computer system to access said executable program on said server computer system;
 - d) establishing direct communication between said client computer system and said digital camera by communicating commands between said client computer system and said digital camera via said executable program;
 - e) accessing via said server computer system data acquired by said digital camera; and
 - f) transferring said data from said digital camera to said client computer system via said server computer system.
23. The method of claim 22 wherein said executable program is a Java servlet.
24. The method of claim 22 wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).
25. A method for allowing a client computer to remotely access a digital image capture device via a communication network wherein a server computer system and said client computer system are communicatively coupled with communication equipment, said method comprising the steps of:
- a) allowing said digital image capture device to establish communication with said server computer system over

27

said communication network, wherein said digital image capture device includes connectivity software that enables said digital image capture device to establish a network connection with said server computer system;

- b) receiving an address of said digital image capture device and registering said address in an executable program on said server computer system, wherein said digital image capture device automatically registers said address with said server computer system;
- c) allowing said client computer system to access said executable program on said server computer system; and

28

- d) establishing direct communication between said client computer system and said digital image capture device by communicating commands between said client computer system and said digital image capture device via said executable program, such that data from said digital image capture device is transferred to said client computer system via said server computer system.

26. The method of claim **25** wherein said executable program is a Java servlet.

27. The method of claim **25** wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).

* * * * *



US00662218B2

(12) **United States Patent**
Mighdoll et al.

(10) Patent No.: **US 6,662,218 B2**
(45) Date of Patent: ***Dec. 9, 2003**

(54) **METHOD OF TRANSCODING DOCUMENTS
IN A NETWORK ENVIRONMENT USING A
PROXY SERVER**

(75) Inventors: **Lee S. Mighdoll**, San Francisco, CA
(US); **Bruce A. Leak**, Palo Alto, CA
(US); **Stephen C. Perlman**, Mountain
View, CA (US); **Phillip Y. Goldman**,
Los Altos, CA (US)

(73) Assignee: **WebTV Networks, Inc.**, Mountain
View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 0 days.

This patent is subject to a terminal dis-
claimer.

(21) Appl. No.: **10/247,885**

(22) Filed: **Sep. 20, 2002**

(65) **Priority Publication Data**

US 2003/0014499 A1 Jan. 16, 2003

Related U.S. Application Data

(63) Continuation of application No. 09/343,067, filed on Jun.
29, 1999, now abandoned, which is a continuation of application
No. 08/656,924, filed on Jun. 3, 1996, now Pat. No.
5,918,013.

(51) **Int. Cl.** **G06F 15/16**

(52) **U.S. Cl.** **709/219; 709/203; 709/217;
709/228; 709/229; 709/246**

(58) **Field of Search** **709/200-203,
709/217-219, 227-229, 231-232, 246-247**

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,575,579 A 3/1986 Simon et al. 178/4
4,852,151 A 7/1989 Dittakavi et al. 379/97

4,922,523 A 5/1990 Hashimoto 379/96
4,975,944 A 12/1990 Cho 379/209
4,995,074 A 2/1991 Goldnan et al. 379/97
5,005,011 A 4/1991 Perlman et al. 340/728
5,095,494 A 3/1992 Takahashi et al. 375/10
5,220,420 A 6/1993 Hoarty et al. 358/86
5,241,587 A 8/1993 Horton et al. 379/92
5,263,084 A 11/1993 Chaput et al. 379/215
5,287,401 A 2/1994 Lin 379/98
5,299,307 A 3/1994 Young 395/161

(List continued on next page.)

OTHER PUBLICATIONS

Rosoff, Matt, Review: "Gateway Destination PC," c/net
inc., 2 pages, Feb. 19, 1996.

Seidman, Robert, Article: "What Larry and Lou Know (That
You Don't)," c/net inc., 2 pages, Jan. 29, 1996.

Stellin, Susan, Article: "The \$500 Web Box: Less is More?"
c/net inc., 2 pages, 1996.

(List continued on next page.)

Primary Examiner—Bharat Barot

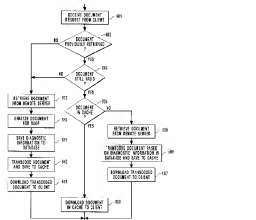
(74) Attorney, Agent, or Firm—Workman, Nydegger

(57)

ABSTRACT

A method is described of providing a document to a client
coupled to a server. The server functions as a proxy on
behalf of the client for purposes of accessing a remote
server. In the method, a document is retrieved from the
remote server in response to a request from the client. The
document includes data to be used by the client in generating
a display. The proxying server alters (i.e., transcodes) the
data in the document to form a transcoded document. The
transcoded document is then transmitted to the client. The
proxying server transcodes the data in the document in order
to perform at least one of the following functions: (1)
matching decompression requirements at the client; (2)
converting the document into a format compatible for the
client; (3) reducing latency experienced by the client; and
(4) altering the document to fit into smaller memory space.

31 Claims, 12 Drawing Sheets



U.S. PATENT DOCUMENTS

5,325,423 A	6/1994	Lewis	379/60	5,678,041 A	10/1997	Baker et al.	395/609
5,329,619 A	7/1994	Page et al.	395/200	5,727,159 A	3/1998	Kikinis	395/200,76
5,341,293 A	8/1994	Verletney et al.	364/419,17	5,842,216 A	11/1998	Anderson et al.	707/203
5,369,688 A	11/1994	Tsukamoto et al.	379/100	5,889,955 A	3/1999	Shinozaki et al.	395/200,54
5,469,540 A	11/1995	Powers, III et al.	395/158	5,918,013 A	* 6/1999	Mighdoli et al.	709/217
5,488,411 A	1/1996	Lewis	348/8	5,978,817 A	11/1999	Gianandrea et al.	707/501
5,490,208 A	2/1996	Remillard	379/96	5,996,022 A	* 11/1999	Krueger et al.	709/247
5,530,852 A	6/1996	Meske, Jr. et al.	395/600	6,018,619 A	1/2000	Allard et al.	395/200,54
5,538,255 A	7/1996	Barker	463/41	6,226,412 B1	5/2001	Schwab	382/232
5,558,339 A	9/1996	Periman	463/42	OTHER PUBLICATIONS			
5,561,709 A	10/1996	Remillard	379/96	<i>Administrator's Guide, Netscape Proxy Server Version 2.0,</i>			
5,564,001 A	10/1996	Lewis	395/154	Netscape Communications Corporation, pp. 19–20, 1996.			
5,572,643 A	11/1996	Judson	395/793	Chankhuthlod, Anawat, et al., "A Hierarchical Internet			
5,586,257 A	12/1996	Periman	463/42	<i>Object Cache,</i> " 1996 USEWIX Technical Conference (6			
5,586,260 A	12/1996	Hu	395/200,2	pages).			
5,612,730 A	3/1997	Lewis	348/8	Farrow, Rik, "Securing the Web: fire walls, proxy servers,			
5,623,600 A	4/1997	Ji et al.	395/187,01	<i>and data driven attacks,</i> " InfoWorld, Jun. 19, 1995, vol. 7,			
5,654,886 A	8/1997	Zereski, Jr. et al.	364/420	No. 25, pp. 103–104.			
5,657,390 A	8/1997	Elgarni et al.	380/49	* cited by examiner			
5,657,450 A	8/1997	Rao et al.	395/610				

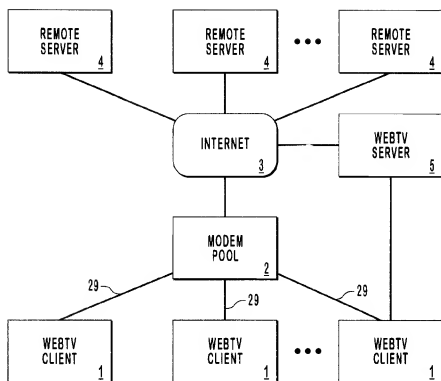


FIG. 1

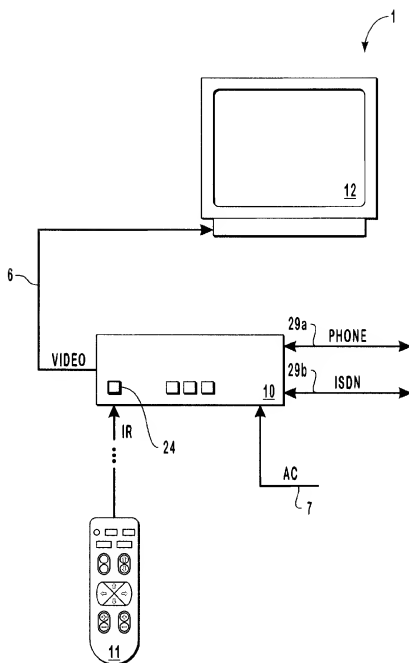


FIG. 2

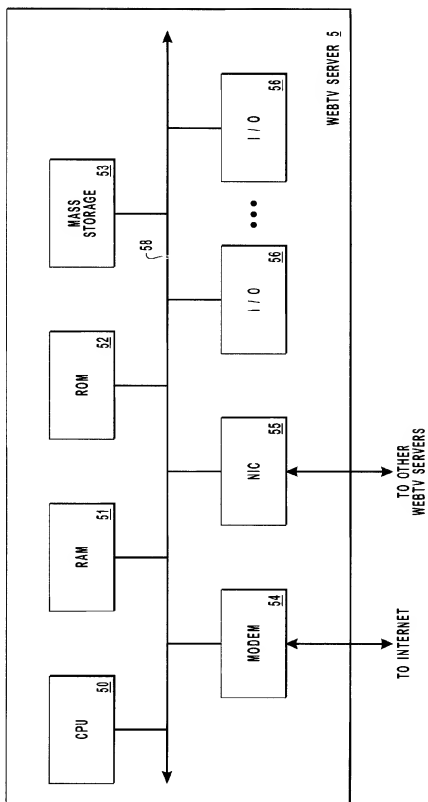


FIG. 3

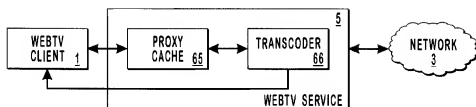


FIG. 4A

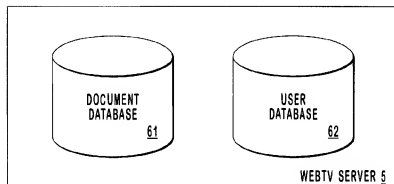


FIG. 4B

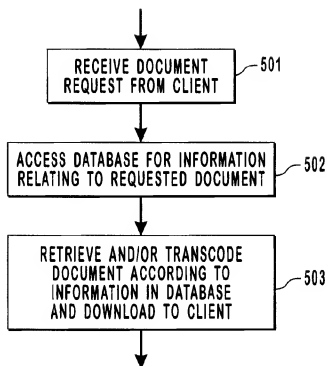


FIG. 5

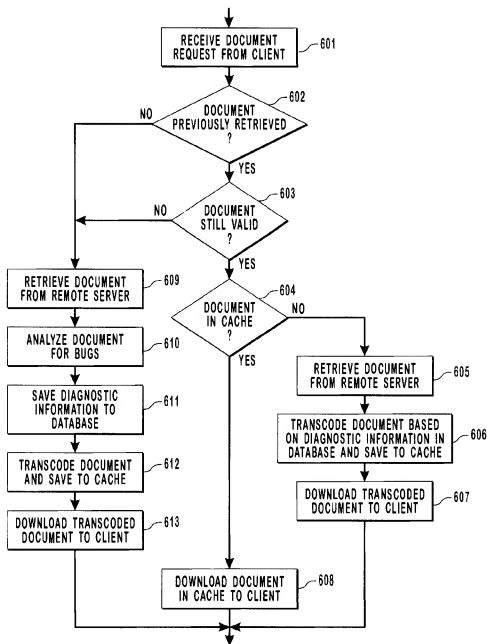


FIG. 6

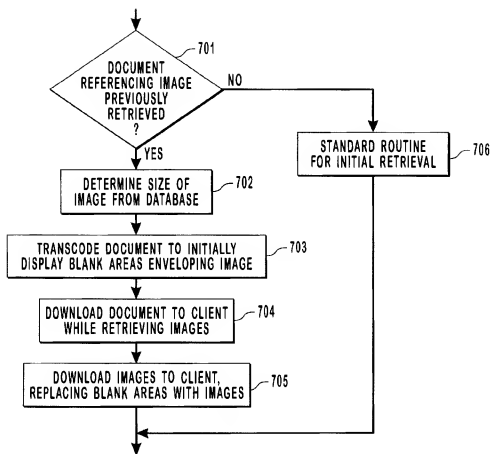


FIG. 7

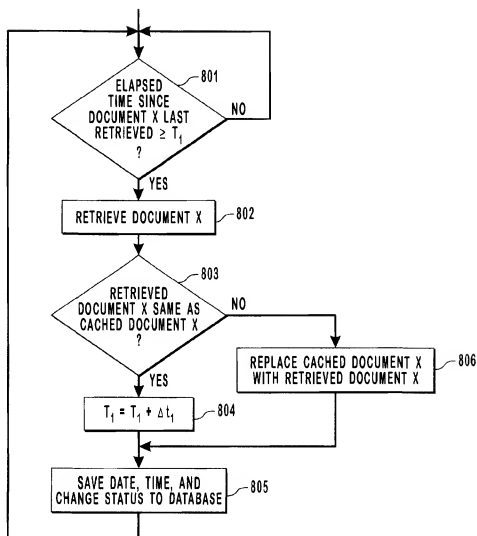


FIG. 8

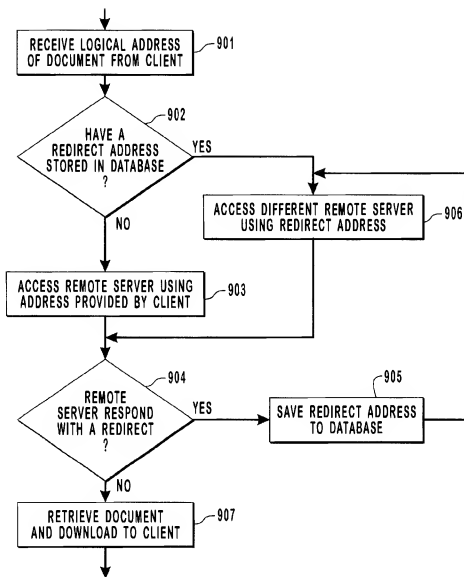


FIG. 9

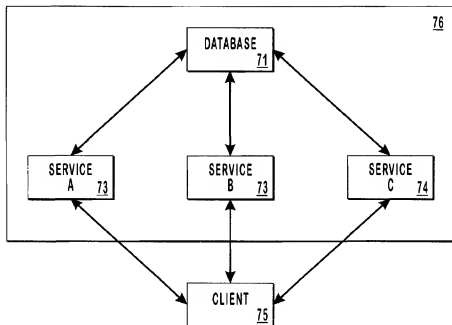


FIG. 10
(PRIOR ART)

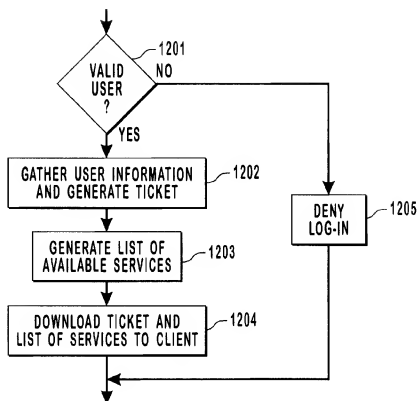


FIG. 12

1

METHOD OF TRANSCODING DOCUMENTS IN A NETWORK ENVIRONMENT USING A PROXY SERVER

RELATED APPLICATION

This is a continuation of U.S. patent application Ser. No. 09/343,067, entitled "Method of Transcoding Documents in a Network Environment Using a Proxy Server," filed Jun. 29, 1999, now abandoned which is a continuation of U.S. application Ser. No. 08/656,924 entitled "Method of Transcoding Documents in a Network Environment Using a Proxy Server," filed Jun. 3, 1996, now U.S. Pat. No. 5,918, 013 both of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. The Field of the Invention

The present invention pertains to the field of client-server computer networking. More particularly, the present invention relates to a method of transcoding documents in a network environment using a proxy server.

2. The Prior State of the Art

The number of people using personal computers has increased substantially in recent years, and along with this increase has come an explosion in the use of the Internet. One particular aspect of the Internet which has gained widespread use is the World-Wide Web ("the Web"). The Web is a collection of formatted hypertext pages located on numerous computers around the world that are logically connected by the Internet. Advances in network technology and software providing user interfaces to the Web ("Web browsers") have made the Web accessible to a large segment of the population. However, despite the growth in the development and use of the Web, many people are still unable to take advantage of this important resource.

Access to the Web has been limited thus far mostly to people who have access to a personal computer. However, many people cannot afford the cost of even a relatively inexpensive personal computer, while others are either unable or unwilling to learn the basic computer skills that are required to access the Web. Furthermore, Web browsers in the prior art generally do not provide the degree of user-friendliness desired by some people, and many computer novices do not have the patience to learn how to use the software. Therefore, it would be desirable to provide an inexpensive means by which a person can access the Web without the use of a personal computer. In particular, it would be desirable for a person to be able to access the Web pages using an ordinary television set and a remote control, so that the person feels more as if he or she is simply changing television channels, rather than utilizing a complex computer network.

Prior art Web technology also has other significant limitations which can make a person's experience unpleasant when browsing the Web. Web documents are commonly written in HTML (Hypertext Mark-up Language). HTML documents sometimes contain bugs (errors) or have features that are not recognized by certain Web browsers. These bugs or quirks in a document can cause a Web browser to fail. Thus, what is needed is a means for reducing the frequency with which client systems fail due to bugs or quirks in HTML documents.

Another problem associated with browsing the Web is latency. People commonly experience long, frustrating delays when browsing the Web. It is not unusual for a person to have to wait minutes after selecting a hypertext link for a

2

Web page to be completely downloaded to his computer and displayed on his computer screen. There are many possible causes for latency, such as heavy communications traffic on the Internet and slow response of remote servers. Latency can also be caused by Web pages including images. One reason for this effect is that, when an HTML document references an image, it takes time to retrieve the image itself after the referencing document has been retrieved. Another reason is that, in the prior art, if the referencing document does not specify the size of the image, the client system generally cannot display the Web page until the image itself has been retrieved. Numerous other sources of latency exist with respect to the Web. Therefore, what is needed is a means for reducing such latency, to eliminate some of the frustration which typically has been associated with browsing the Web.

Security is another concern associated with the Internet. Internet service providers (ISPs) generally maintain certain information about each customer in a database. This information may include information which a customer may not wish to become publicly known, such as social security numbers and credit card numbers. Maintaining the confidentiality of this information in a system that is connected to an expensive publicly-accessible computer network like the Internet can be problematic. Further, the problem can be aggravated by the fact that an ISP often provides numerous different services, each of which has access to this database. Allowing access to the database by many different entities creates many opportunities for security breaches to occur. Therefore, what is needed is a way to improve the security of confidential customer information in a server system coupled to the Internet.

SUMMARY AND OBJECTS OF THE INVENTION

A method is described of providing a document to a client coupled to a server. The server functions as a proxy on behalf of the client for purposes of accessing a remote server. In the method, a document is retrieved from the remote server in response to a request from the client. The document includes data to be used by the client in generating a display. The proxying server alters (i.e., transcodes) the data in the document to form a transcoded document. The transcoded document is then transmitted to the client.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follows. The proxying server transcodes the data in the document in order to perform at least one of the following functions: (1) matching decompression requirements at the client; (2) converting the document into a format compatible for the client; (3) reducing latency experienced by the client; and (4) altering the document to fit into smaller memory space.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follow.

BRIEF DESCRIPTION OF THE DRAWINGS

In order that the manner in which the above-recited and other advantages and objects of the invention are obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered limiting of its scope, the invention will be

3

described and explained with additional specificity and detail through the use of the accompanying drawings in which:

FIG. 1 illustrates several clients connected to a proxying server in a network;

FIG. 2 illustrates a client according to the present invention;

FIG. 3 is a block diagram of a server according to the present invention;

FIG. 4A illustrates a server including a proxy cache and a transcoder;

FIG. 4B illustrates databases used in a server according to the present invention;

FIG. 5 is a flow diagram illustrating a routine for transcoding a document retrieved from a remote server using data stored in a persistent database;

FIG. 6 is a flow diagram illustrating a routine for transcoding an HTML document for purposes of eliminating bugs or undesirable features;

FIG. 7 is a flow diagram illustrating a routine for reducing latency when downloading a document referencing an image to a client;

FIG. 8 is a flow diagram illustrating a routine for updating documents stored in the proxy cache using data stored in a persistent database;

FIG. 9 is a flow diagram illustrating a routine used by a server for retrieving documents from another remote server;

FIG. 10 is a block diagram of a prior art server system showing a relationship between various services and a database;

FIG. 11 is a block diagram of a server system according to the present invention showing a relationship between various services and a user database; and

FIG. 12 is a flow diagram illustrating a routine used by a server for regulating access to various services provided by the server.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

A method and apparatus are described for providing proxying and transcoding of documents in a network. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be evident, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

The present invention includes various steps, which will be described below. The steps can be embodied in machine-executable instructions, which can be used to cause a general-purpose or special-purpose processor programmed with the instructions to perform the steps. Alternatively, the steps of the present invention might be performed by specific hardware components that contain hardwired logic for performing the steps, or by any combination of programmed computer components and custom hardware components.

I. System Overview

The present invention is included in a system, known as WebTV™, for providing a user with access to the Internet. A user of a WebTV™ client generally accesses a WebTV™ server via a direct-dial telephone (POTS, for "plain old

4

telephone service"), ISDN (Integrated Services Digital Network), or other similar connection, in order to browse the Web, send and receive electronic mail (e-mail), and use various other WebTV™ network services. The WebTV™ network services are provided by WebTV™ servers using software residing within the WebTV™ servers in conjunction with software residing within a WebTV™ client.

FIG. 1 illustrates a basic configuration of the WebTV™ network according to one embodiment. A number of WebTV™ clients 1 are coupled to a modem pool 2 via direct-dial, bi-directional data connections 29, which may be telephone (POTS, i.e., "plain old telephone service"), ISDN (Integrated Services Digital Network), or any other similar type of connection. The modem pool 2 is coupled typically through a router, such as that conventionally known in the art, to a number of remote servers 4 via a conventional network infrastructure 3, such as the Internet. The WebTV™ system also includes a WebTV™ server 5, which specifically supports the WebTV™ clients 1. The WebTV™ clients 1 each have a connection to the WebTV™ server 5 either directly or through the modem pool 2 and the Internet 3. Note that the modem pool 2 is a conventional modem pool, such as those found today throughout the world providing access to the Internet and private networks.

Note that in this description, in order to facilitate explanation the WebTV™ server 5 is generally discussed as if it were a single device, and functions provided by the WebTV™ services are generally discussed as being performed by such single device. However, the WebTV™ server 5 may actually comprise multiple physical and logical devices connected in a distributed architecture, and the various functions discussed below which are provided by the WebTV™ services may actually be distributed among multiple WebTV™ server devices.

II. Client System

FIG. 2 illustrates a WebTV™ client 1. The WebTV™ client 1 includes an electronics unit 10 (hereinafter referred to as "the WebTV™ box 10"), an ordinary television set 12, and a remote control 11. In an alternative embodiment of the present invention, the WebTV™ box 10 is built into the television set 12 as an integral unit. The WebTV™ box 10 includes hardware and software for providing the user with a graphical user interface, by which the user can access the WebTV™ network services, browse the Web, send e-mail, and otherwise access the Internet.

The WebTV™ client 1 uses the television set 12 as a display device. The WebTV™ box 10 is coupled to the television set 12 by a video link 6. The video link 6 is an RF (radio frequency), S-video, composite video, or other equivalent form of video link. In the preferred embodiment, the client 1 includes both a standard modem and an ISDN modem, such that the communication link 29 between the WebTV™ box 10 and the server 5 can be either a telephone (POTS) connection 29a or an ISDN connection 29b. The WebTV™ box 10 receives power through a power line 7.

Remote control 11 is operated by the user in order to control the WebTV™ client 1 in browsing the Web, sending e-mail, and performing other Internet-related functions. The WebTV™ box 10 receives commands from remote control 11 via an infrared (IR) communication link. In alternative embodiments, the link between the remote control 11 and the WebTV™ box 10 may be RF or any equivalent mode of transmission.

III. Server System

The WebTV™ server 5 generally includes one or more computer systems generally having the architecture illus-

5

trated in FIG. 3. It should be noted that the illustrated architecture is only exemplary, the present invention is not constrained to this particular architecture. The illustrated architecture includes a central processing unit (CPU) 50, random access memory (RAM) 51, read-only memory (ROM) 52, a mass storage device 53, a modem 54, a network interface card (NIC) 55, and various other input/output (I/O) devices 56. Mass storage device 53 includes a magnetic, optical, or other equivalent storage medium. I/O devices 56 may include any or all of devices such as a display monitor, keyboard, cursor control device, etc. Modem 54 is used to communicate data to and from remote servers 4 via the Internet.

As noted above, the WebTV™ server 5 may actually comprise multiple physical and logical devices connected in a distributed architecture. Accordingly, NIC 55 is used to provide data communication with other devices that are part of the WebTV™ services. Modem 54 may also be used to communicate with other devices that are part of the WebTV™ services and which are not located in close geographic proximity to the illustrated device.

According to the present invention, the WebTV™ server 5 acts as a proxy in providing the WebTV™ client 1 with access to the Web and other WebTV™ services. More specifically, WebTV™ server 5 functions as a "caching proxy." FIG. 4A illustrates the caching feature of the WebTV™ server 5. In FIG. 4A, the WebTV™ server 5 is functionally located between the WebTV™ client 1 and the Internet infrastructure 3. The WebTV™ server 5 includes a proxy cache 65 which is functionally coupled to the WebTV™ client 1. The proxy cache 65 is used for temporary storage of Web documents, images, and other information which is frequently used by either the WebTV™ client 1 or the WebTV™ server 5.

A document transcoder 66 is functionally coupled between the proxy cache 65 and the Internet infrastructure 3. The document transcoder 66 includes software which is used to automatically revise the code of Web documents retrieved from the remote servers 4, for purposes which are described below.

The WebTV™ service provides a document database 61 and a user database 62, as illustrated in FIG. 4B. The user database 62 contains information that is used to control certain features relating to access privileges and capabilities of the user of the client 1. This information is used to regulate initial access to the WebTV™ service, as well as to regulate access to the individual services provided by the WebTV™ system, as will be described below. The document database 61 is a persistent database which stores certain diagnostic and historical information about each document and image retrieved by the server 5, as is now described.

A. Document Database

The basic purpose of the document database 61 is that, after a document has once been retrieved by the server 5, the stored information can be used by the server 5 to speed up processing and downloading of that document in response to all future requests for that document. In addition, the transcoding functions and various other functions of the WebTV™ service are facilitated by making use of the information stored in the document database 61, as will be described below.

Referring now to FIG. 5, the server 5 initially receives a document request from a client 1 (step 501). The document request will generally result from the user of the client 1

6

activating a hypertext anchor (link) on a Web page. The act of activating a hypertext anchor may consist of clicking on underlined text in a displayed Web page using a mouse, for example. The document request will typically (but not always) include the URL (Uniform Resource Locator) or other address of the selected anchor. Upon receiving the document request, the server 5 optionally accesses the document database 61 to retrieve stored information relating to the requested document (step 502). It should be noted that the document database 61 is not necessarily accessed in every case. The information retrieved from the document database 61 is used by the server 5 for determining, among other things, how long a requested document has been cached and/or whether the document is still valid. The criteria for determining validity of the stored document are discussed below. The server 5 retrieves the document from the cache 65 if the stored document is valid; otherwise, the server 5 retrieves the document from the appropriate remote server 4 (step 503). The server 5 automatically transcodes the document as necessary based on the information stored in the document database 61 (step 503). The transcoding functions are discussed further below.

The document database 61 includes certain historical and diagnostic information for every Web page that is accessed at any time by a WebTV™ client 1. As is well known, a Web page may correspond to a document written in a language such as HTML (Hypertext Mark-Up Language), VRML (Virtual Reality Modelling Language), or another suitable language. Alternatively, a Web page may represent an image, or a document which references one or more images. According to the present invention, once a document or image is retrieved by the WebTV™ server 5 from a remote server 4 for the first time, detailed information on this document or image is stored permanently in the document database 61. More specifically, for every Web page that is retrieved from a remote server 4, any or all of the following data are stored in the document database 61:

- 1) information identifying bugs (errors) or quirks in the Web page, or undesirable effects caused when the Web page is displayed by a client 1;
- 2) relevant bug-finding algorithms;
- 3) the date and time the Web page was last retrieved;
- 4) the date and time the Web page was most recently altered by the author;
- 5) a checksum for determining whether the Web page has been altered;
- 6) the size of the Web page (in terms of memory);
- 7) the type of Web page (e.g., HTML document, image, etc.);
- 8) a list of hypertext anchors (links) in the Web page and corresponding URLs;
- 9) a list of the most popular anchors based on the number of "hits" (requests from a client 1);
- 10) a list of related Web pages which can be prefetched;
- 11) whether the Web page has been redirected to another remote server 4;
- 12) a redirect address (if appropriate);
- 13) whether the redirect (if any) is temporary or permanent, and if permanent, the duration of the redirect;
- 14) if the Web page is an image, the size of the image in terms of both physical dimensions and memory space;
- 15) the sizes of in-line images (images displayed in text) referenced by the document defining the Web page;

7

- 16) the size of the largest image referenced by the document;
- 17) information identifying any image maps in the Web page;
- 18) whether to resize any images corresponding to the Web page;
- 19) an indication of any forms or tables in the Web page;
- 20) any unknown protocols;
- 21) any links to "dead" Web pages (i.e., pages which are no longer active);
- 22) the latency and throughput of the remote server 4 on which the Web page is located;
- 23) the character set of the document;
- 24) the vendor of the remote server 4 on which the Web page is located;
- 25) the geographic location of the remote server 4 on which the Web page is located;
- 26) the number of other Web pages which reference the subject Web page;
- 27) the compression algorithm used by the image or document;
- 28) the compression algorithm chosen by the transcoder;
- 29) a value indicating the popularity of the Web page based on the number of hits by clients; and
- 30) a value indicating the popularity of other Web pages which reference the subject Web page.

B. Transcoding

As mentioned above, the WebTV™ services provide a transcoder 66, which is used to rewrite certain portions of the code in an HTML document for various purposes. These purposes include: (1) correcting bugs in documents; (2) correcting undesirable effects which occur when a document is displayed by the client 1; (3) improving the efficiency of transmission of documents from the server 5 to the client 1; (4) matching hardware decompression technology within the client 1; (5) resizing images to fit on the television set 12; (6) converting documents into other formats to provide compatibility; (7) reducing latency experienced by a client 1 when displaying a Web page with in-line images (images displayed in text); and, (8) altering documents to fit into smaller memory spaces.

There are three transcoding modes used by the transcoder 66: (1) streaming, (2) buffered, and (3) deferred. Streaming transcoding refers to the transcoding of documents on a line-by-line basis as they are retrieved from a remote server 4 and downloaded to the client 1 (i.e., transcoding "on the fly"). Some documents, however, must first be buffered in the WebTV™ server 5 before transcoding and downloading them to the client 1. A document may need to be buffered before transmitting it to the client 1 if the type of changes to be made can only be made after the entire document has been retrieved from the remote server 4. Because the process of retrieving and downloading a document to the client 1 increases latency and decreases throughput, it is not desirable to buffer all documents. Therefore, the transcoder 66 accesses and uses information in the document database 61 relating to the requested document to first determine whether a requested document must be buffered for purposes of transcoding, before the document is retrieved from the remote server 4.

In the deferred mode, transcoding is deferred until after a requested document has been downloaded to a client 1. The deferred mode therefore reduces latency experienced by the

8

client 1 in receiving the document. Transcoding may be performed immediately after downloading or any time thereafter. For example, it may be convenient to perform transcoding during periods of low usage of WebTV™ services, such as at night. This mode is useful for certain types of transcoding which are not mandatory.

1. Transcoding for Bugs and Quirks

One characteristic of some prior art Web browsers is that they may experience failures ("crashes") because of bugs or unexpected features ("quirks") that are present in a Web document. Alternatively, quirks in a document may cause an undesirable result, even though the client does not crash. Therefore, the transcoding feature of the present invention provides a means for correcting certain bugs and quirks in a Web document. To be corrected by the transcoder 66, bugs and quirks must be identifiable by software running on the server 5. Consequently, the transcoder 66 will generally only correct conditions which have been previously discovered, such as those discovered during testing or reported by users. Once a bug or quirk is discovered, however, algorithms are added to the transcoder 66 to both detect the bug or quirk in the future in any Web document and to automatically correct it.

There are countless possibilities of bugs or quirks which might be encountered in a Web document. Therefore, no attempt will be made herein to provide an exhaustive list. Nonetheless, some examples may be useful at this point. Consider, for example, an HTML document that is downloaded from a remote server 4 and which contains a table having a width specified in the document as "0." This condition might cause a failure if the client were to attempt to display the document as written. This situation therefore, can be detected and corrected by the transcoder 66. Another example is a quirk in the document which causes quotations to be terminated with too many quotation marks. Once the quirk is first detected and an algorithm is written to recognize it, the transcoder 66 can automatically correct the quirk in any document.

If a given Web document has previously been retrieved by the server 5, there will be information regarding that document available in the document database 61 as described above. The information regarding this document will include whether or not the document included any bugs or quirks that required transcoding when the document was previously retrieved. The transcoder 66 utilizes this information to determine whether (1) the document is free of bugs and quirks, (2) the document has bugs or quirks which can be remedied by transcoding on the fly, or (3) the document has bugs or quirks which cannot be corrected on the fly (i.e., buffering is required).

FIG. 6 illustrates a routine for transcoding a Web document for purposes of eliminating bugs and quirks. Initially, the server 5 receives a document request from the client 1 (step 601). Next, the document database 61 is accessed to determine whether or not the requested document has been previously retrieved (step 602). If the document has not been previously retrieved, then the server 5 retrieves the document from the remote server 4 (step 609). Next, the retrieved document is analyzed for the presence of bugs or unusual conditions (step 610). Various diagnostic information is then stored in the document database 61 as a result of the analysis to note any bugs or quirks that were found (step 611). If any bugs or quirks were found which can be corrected by the transcoder 66, the document is then transcoded and saved to the proxy cache 65 (step 612). The transcoded document is

9

then downloaded to the client 1 (step 613). It should be noted that transcoding can be deferred until after the document has been downloaded, as described above; hence, the sequence of FIG. 6 is illustrative only.

If (in step 602) the requested document had been previously retrieved, then it is determined whether the requested document is still valid (step 603) and whether the document is present in the proxy cache 65 (step 604). If the document is no longer valid, then the document is retrieved from the remote server 4, analyzed for bugs and quirks, transcoded as required, and then downloaded to the client 1 as described above (steps 609-613). Methods for determining validity of a document are discussed below. If the document is still valid (step 603) and the document is present in the cache 65, the document is downloaded to the client 1 in its current form (as it is stored in the cache), since it has already been transcoded (step 608).

The document, however, may be valid but not present in the cache. This may be the case, for example, if the document has not been requested recently and the cache 65 has become too full to retain the requested document. In that case, the document is retrieved again from the remote server 4 (step 605) and then transcoded on the basis of the previously-acquired diagnostic information stored within the database 61 for that document. The document is then saved to the cache 65 (step 606). Note that because the document is still valid, it is assumed that the diagnostic information stored in the document database 61 for that document is still valid and that the transcoding can be performed on the basis of that information. Accordingly, once the document is transcoded, the transcoded document is downloaded to the client 1 (step 607). Again, note that transcoding can be deferred until after the document has been downloaded in some cases.

The validity of the requested document can be determined based on various different criteria. For example, some HTML documents specify a date on which the document was created, a length of time for which the document will be valid, or both. The validity determination can be based upon such information. For example, a document which specifies only the date of creation can be automatically deemed invalid after a predetermined period of time has passed.

Alternatively, validity can be based upon the popularity of the requested document. "Popularity" can be quantified based upon the number of hits for that document, which is tracked in the document database 61. For example, it might be prudent to simply assign a relatively short period of validity to a document which is very popular and a longer period of validity to a document which is less popular.

Another alternative basis for the validity of a document is the observed rate of change of the document. Again, data in the persistent document database 61 can be used. That is, because the document database 61 stores the date and time on which the document was last observed to change, the server 5 can approximate how often the document actually changes. A document or image which is observed to change frequently (e.g., a weather map or a news page) can be assigned a relatively short period of validity. It will be recognized that numerous other ways of determining validity are possible.

2. Transcoding to Reduce Latency

Another purpose for transcoding is to allow documents requested by a client 1 to be displayed by the client 1 more rapidly. Many HTML documents contain references to "in-line" images, or images that will be displayed in text in a

10

Web page. The normal process used in the prior art to display a Web page having in-line images is that the HTML document referencing the image is first downloaded to the client, followed by the client's requesting the referenced image. The referenced image is then retrieved from the remote server on which it is located and downloaded to the client. One problem associated with the prior art, however, is that the speed with which a complete Web page can be displayed to the user is often limited by the time it takes to retrieve in-line images. One reason for this is that it simply takes time to retrieve the image itself after the referencing document has been retrieved. Another reason is that, in the prior art, if the referencing document does not specify the size of the image, the Web page generally cannot be displayed until the image itself has been retrieved. The present invention overcomes these limitations.

According to the present invention, information stored in the document database 61 regarding the in-line images is used to transcode the referencing document in order to reduce latency in displaying the Web page. Once any document which references an in-line image is initially retrieved by the server 5, the fact that the document references an in-line image is stored in the document database 61. In addition, the size of the image is determined, either from the document (if specified) or from the image itself, and then stored in the document database 61. Consequently, for documents which do not specify the size of their in-line images, the size information stored in the database 61 is then used the next time the document is requested in order to reduce latency in downloading and displaying the Web page.

Refer now to FIG. 7, which illustrates a routine for reducing latency when downloading a document referencing an image to a client 1. Assume that a client 1 sends a request to the server 5 for an HTML document containing a reference to an in-line image. Assume further that the size of the image is not specified in the document itself. Initially, the server 5 determines whether that document has been previously retrieved (step 701). If not, the standard initial retrieval and transcoding procedure is followed (step 706), as described in connection with FIG. 6. If, however, the document has been previously retrieved, then the transcoder 66 accesses the size information stored in the document database 61 for the in-line image (step 702). Based on this size information, the HTML document is transcoded such that, when the Web page is initially displayed by the client 1, the area in which the image belongs is replaced by a blank region enveloping the shape of the image (step 703). Thus, any in-line image referenced by a document is displayed initially as a blank region. Consequently, the client 1 can immediately display the Web page corresponding to the HTML document even before the referenced image has been retrieved or downloaded (i.e., even before the size of the image is known to the client 1).

As the transcoded HTML document is downloaded to the client, the image is retrieved from the appropriate remote server 4 (step 704). Once the image is retrieved from the remote server 4 and downloaded to the client 1, the client 1 replaces the blank area in the Web page with the actual image (step 705).

3. Transcoding to Display Web Pages on a Television

As noted above, the client 1 utilizes an ordinary television set 12 as a display device. However, images in Web pages are generally formatted for display on a computer monitor, not a television set. Consequently, the transcoding function

11

of the present invention is used to resize images for display on the television set 12. This includes resealing images as necessary to avoid truncation when displayed on the television set 12.

It should be noted that prior art Web browsers which operate on computer monitors typically use resizable windows. Hence, the size of the visible region varies from client to client. However, because the web browser used by the WebTV™ client 1 is specifically designed for display on a television set, the present invention allows documents and images to be formatted when they are cached.

4. Transcoding for Transmission Efficiency

Documents retrieved by the server 5 are also transcoded to improve transmission efficiency. In particular, documents can be transcoded in order to reduce high frequency components in order to reduce interface flicker when they are displayed on a television set. Various methods for coding software or hardware to reduce perceptual interface flicker are described in co-pending U.S. patent application Ser. No. 08/656,923, filed on Jun. 3, 1996.

Documents can also be transcoded in order to lower the resolution of the displayed Web page. Reducing the resolution is desirable, because images formatted for computer systems will generally have a higher resolution than the NTSC (National Television Standards Committee) video format used by conventional television sets. Since the NTSC video does not have the bandwidth to reproduce the resolution of computer-formatted images, the bandwidth consumed in transmitting images to the client 1 at such a high resolution would be wasted.

5. Other Uses for Transcoding

Transcoding is also used by the present invention to recode a document using new formats into older, compatible formats. Images are often displayed in the JPEG (Joint Picture Experts Group) format or the GIF image format. JPEG often consumes less bandwidth than GIF, however. Consequently, images which are retrieved in GIF format are sometimes transcoded into JPEG format. Methods for generally converting images between GIF and JPEG formats are well known.

Other uses for transcoding include transcoding audio files. For example, audio may be transcoded into different formats in order to achieve a desired balance between memory usage, sound quality, and data transfer rate. In addition, audio may be transcoded from a file format (e.g., an ".AU" file) to a streaming format (e.g., MPEG 1 audio). Yet another use of audio transcoding is the transcoding of MIDI (Musical Instrument Digital Interface) data to streaming variants of MIDI.

Additionally, documents or images requiring a large amount of memory (e.g., long lists) can be transcoded in order to consume less memory space in the client 1. This may involve, for example, separating a large document or image into multiple sections. For example, the server 5 can insert tags at appropriate locations in the original document so that the document appears to the client 1 as multiple Web pages. Hence, while viewing a given page representing a portion of the original document, the user can view the next page (i.e., the next portion of the original document) by activating a button on the screen as if it were an ordinary hypertext anchor.

C. Proxying

As noted above, the server 5 functions as a proxy on behalf of the client 1 for purposes of accessing the Web. The

12

document database 61 is used in various ways to facilitate this proxy role, as will now be described.

1. Updating Cached Documents

It is desirable to store frequently-requested HTML documents and images in the proxy cache 65 to further reduce latency in providing Web pages to the client 1. However, because some documents and images change over time, documents in the cache 65 will not be valid indefinitely, as mentioned above. A weather map or a news-related Web page, for example, are likely to be updated quite frequently. Consequently, it is desirable for the server 5 to have the ability to estimate the frequency with which documents change, in order to determine how long a document can safely remain within the proxy cache 65 without being updated.

The persistent database 65 is used to store the date and time of the last several fetches of each document and image retrieved from a remote server 4, along with an indication of any changes that were detected, if any. A document or image which has been stored in the cache 65 is then retrieved on a periodic basis to determine if it has been changed. Change status information indicating whether the document has changed since the previous fetch is then stored in the document database 61. If no changes are detected, then the time interval between fetches of this document is increased. If the document has changed, the time interval is maintained or decreased. As a result, items in the cache 65 which change frequently will be automatically updated at frequent intervals, whereas documents which do not change often will be replaced in the cache less frequently.

FIG. 8 illustrates a routine for updating documents stored in the proxy cache 65 using data stored in the document database 61. Assume a document X has been stored in the proxy cache 65. Document X remains in the cache 65 until a predetermined update period T_1 expires (step 801). Upon the expiration of the update period T_1 , the document X is again retrieved from the appropriate remote server 4 (step 802). The newly-retrieved document X is then compared to the cached version of document X (step 803). If the document has changed, then the cached version of document X is replaced with the newly-retrieved version of document X (step 806). If not, then the update period T_1 is increased according to a predetermined time increment ΔT_1 (step 804). In any case, the date and time and the change status of document X is saved to the document database 61 (step 805).

2. Document and Image Prefetching

The document database 61 is also used by the server 5 to store prefetching information relating to documents and images. In particular, the database stores, for each document that has been retrieved, a list of images referenced by the document, if any, and their locations. Consequently, the next time a document is requested by a client 1, the images can be immediately retrieved by the server 5 (from the cache 65, if available, or from the remote server 4), even before the client 1 requests them. This procedure improves the speed with which requested Web pages are downloaded to the client.

The document database 61 is also used to facilitate a process referred to as "server-advised client prefetching." Server-advised client prefetching allows the server 5 to inform the client 1 of documents or images which are popular to allow the client 1 to perform the prefetching. In particular, for any given document, a list is maintained in the

13

server 5 of the most popular hypertext anchors in that document (i.e., those which have previously received a large number of hits). When that document is requested by the client 1, the server 5 provides the client 1 with an indication of these popular links.

3. Redirects

Web pages are sometimes forwarded from the remote server on which they are initially placed to a different location. Under the HTTP (Hypertext Transport Protocol), such forwarding is sometimes referred to as a "redirect." When an HTML document is initially stored on one remote server and then later transferred to another remote server, the first remote server will provide, in response to a request for that document, an indication that the document has been transferred to a new remote server. This indication generally includes a forwarding address ("redirect address"), which is generally a URL.

In the prior art, when a computer requesting a Web page receives a redirect, it must then submit a new request to the redirect address. Having to submit a second request and wait for a second response consumes time and increases overall latency. Consequently, the present invention uses the document database 61 to store any redirect address for each document or image. Any time a redirected document is requested, the server 5 automatically accesses the redirect address to retrieve the document. The document or image is provided to the client 1 based on only a single request from the client 1. The change in location of the redirected document or image remains completely transparent to the client 1.

FIG. 9 illustrates a routine performed by the server 5 in accessing documents which may have been forwarded to a new remote server. Initially, the server 5 receives a request for a document, which generally includes an address (step 901). The server 5 then accesses the document database 61 to determine whether there is a redirect address for the requested document (step 902). If there is no redirect address, then the server 5 accesses a remote server 4 based on the address provided in the document request from the client 1 (step 903). Assuming that the remote server 4 does not respond to the server 5 with a redirect (step 904), the document is retrieved and downloaded to the client 1 by the server 5 (step 907). If, however, a redirect address was stored in the document database 61 (step 902), then the server 5 accesses the requested document according to the redirect address (step 906). Or, if the remote server 4 responded with a redirect (step 904), then the server 5 saves the redirect address to the document database 61 (step 905) and accesses the requested document according to the redirect address (step 906).

4. Other Proxy Functions

The document database 61 also stores information relating to the performance of each remote server 4 from which a document is retrieved. This information includes the latency and throughput of the remote server 4. Such information can be valuable in instances where a remote server 4 has a history of responding slowly. For example, when the document is requested, this knowledge can be used by the server 5 to provide a predefined signal to the client 1. The client 1 can, in response to the signal, indicate to the user that a delay is likely and give the user the option of canceling the request.

5. Backoff Mode

Although the server 5 generally operates in the proxy mode, it can also enter a "backoff mode" in which the server

14

5 does not act as a proxy, or the server 5 performs only certain aspects of the normal proxying functions. For example, if the proxy cache 65 is overloaded, then the server 5 can enter a backoff mode in which documents are not cached but are transcoded as required. Alternatively, during times when the server 5 is severely overloaded with network traffic, the server 5 may instruct the client 1 to bypass the server 5 and contact remote servers 4 directly for a specified time or until further notice. Or, the server 5 can enter a flexible backoff mode in which the client 1 will be instructed to contact a remote server 4 directly only for certain Web sites for a limited period of time.

D. Access to WebTV™ Services

The WebTV™ server 5 provides various services to the client 1, such as proxying and electronic mail ("e-mail"). In the prior art, certain difficulties are associated with allowing a client computer access to different services of an Internet service, as will now be explained with reference to FIG. 10.

FIG. 10 illustrates a client-server system according to one prior art embodiment. The server 76 provides various services A, B, and C. The server 76 includes a database 71 for storing information on the user's access privileges to services A, B, and C. The client 75 of the embodiment of FIG. 10 accesses any of services A, B, and C by contacting that service directly. The contacted service then accesses the database 71, which stores the access privileges of the client 75, to determine whether the client 75 should be allowed to access that service. Hence, each service provided by the server 76 requires direct access to the database 71. This architecture results in a large number of accesses being made to the database 71, which is undesirable. In addition, the fact that each service independently has access to the database 71 raises security concerns. Specifically, it can be difficult to isolate sensitive user information. The present invention overcomes such difficulties using a technique which is now described.

1. Tickets Containing Privileges And Capabilities

As shown in FIG. 11, the server 5 provides a number of services D, E, and F 77, 79, and 80, respectively, and a log-in service 78. The log-in service 78 is used specifically to control initial log-on procedures by a client 1. The log-in service 78 has exclusive access to the user database 62 (discussed above with respect to FIG. 4B). The log-in service 78 and the user database 62 are located within a first security zone 84. Service D is located within a second security zone 86, while services E and F are contained within a third security zone 88. Note that the specific arrangement of security zones 84, 86, and 88 with respect to services D, E, and F is illustrative only.

The user database 62 of the present invention stores various information pertaining to each authorized user of a client 1. This information includes account information, a list of the WebTV™ services that are available to the particular user, and certain user preferences. For example, a particular user may not wish his client 1 to be used to access Web pages having adult-oriented subject matter. Consequently, the user would request that his account be filtered to prevent access to such material. This request would then be stored as part of the user data in the user database 62.

With regard to user preferences, the hypertext links selected by a given user can be tracked, and those having the largest number can be stored in the user database 62. The list can then be provided to the client 1 for use in generating a

15

menu screen of the user's favorite Web sites, to allow the user to directly access those Web sites. The list can also be used by the server 5 to analyze the user's interests and to formulate and provide to the user a list of new Web sites which the user is likely to be interested in. The list might be composed by associated key words in Web pages selected by the user with other Web pages.

Referring again to FIG. 11, in response to a log-on request by a client 1, the log-in service 78 consults the user database 62 to determine if access to the server 5 by this particular client 1 is authorized. Assuming access is authorized, the log-in service 78 retrieves certain user information pertaining to this particular client 1 from the user database 62. The log-in service then generates a "ticket" 82, which is an information packet including the retrieved information. The ticket 82 is then provided to the client 1 which requested access.

The ticket 82 includes all information necessary to describe the access privileges of a particular user with respect to all services provided by the server 5. For example, the ticket may include the user name registered to the client 1, the e-mail address assigned to client 1, and any filtering requested by the user with respect to viewing Web sites. Each time the user requests access to one of the services D, E, or F, the client 1 submits a copy of the ticket 82 to that service. The requested service can then determine from the copy of the ticket 82 whether access to that service by that client 1 is authorized and, if so, any important information relating to such access.

None of the services provided by the server 5, other than the log-in service 78, has access to the user database 62. Hence, any security-sensitive information can be isolated within the user database 62 and the log-in service 78. Such isolation allows the individual services provided by the server 5 to be placed within separate "firewalls" (security regions), illustrated as security zones 84, 86, and 88. In addition, this technique greatly reduces the number of accesses required to the user database 62 compared to the prior art embodiment illustrated in FIG. 10.

2. Redundancy of Services and Load Balancing

The present invention also includes certain redundancies in the various services provided by the server 5. In particular, a given service (e.g., e-mail) can be provided by more than one physical or logical device. Each such device is considered a "provider" of that service. If a given provider is overloaded, or if the client 1 is unable to contact that provider, the client 1 can contact any of the other providers of that service. When the server 5 receives a log-in request from a client 1, in addition to generating the above-described ticket 82, the log-in service 78 dynamically generates a list of available WebTV™ services and provides this list to the client 1.

The server 5 can update the list of services used by any client 1 to reflect services becoming unavailable or services coming on-line. Also, the list of services provided to each client 1 can be updated by the server 5 based upon changes in the loading of the server 5, in order to optimize traffic on the server 5. In addition, a client's list of services can be updated by services other than the log-in service 78, such that one service can effectively introduce another service to the client 1. For example, the e-mail service may provide a client 1 with the name, port number and IP of its address book service. Thus, one service can effectively, and securely within the same chain of trust, introduce another service to the client 1.

16

This list of services includes the name of each service, a port number for the provider of each service, and an IP (Internet Protocol) for each service. Different providers of the same service are designated by the same name, but different port numbers and/or IPs. Note that in a standard URL, the protocol is normally specified at the beginning of the URL, such as "HTTP://www. . . ." under the HTTP protocol. However, according to the present invention, the normal protocol designation (i.e., "HTTP") in the URL is replaced with the name of the service, since the port number and IP for each service are known to the client 1. Hence, the client 1 can access any of the redundant providers of a given service using the same URL. This procedure effectively adds a level of indirection to all accesses made to any WebTV™ service and automatically adds redundancy to the proxy service. It should also be noted that separate service names can also refer to the same service.

Assume, for example, that the e-mail service provided by the WebTV™ system is designated by the service name "WTV-mailto." A client 1 can access any provider of this e-mail service using the same URL. The client 1 merely chooses the appropriate port number and IP number to distinguish between providers. If the client 1 is unable to connect to one e-mail provider, it can simply contact the next one in the list.

Thus, at log-in time, a client 1 is provided with both a ticket containing privileges and capabilities as well as a list of service providers, as illustrated in FIG. 12. Initially, the log-in service 78 determines whether the user of client 1 is a valid user (step 1201). If not, log-in is denied (step 1205). If the user is a valid user, then the log-in service 78 gathers user information from the user database 62 and generates a ticket 82 (step 1202). The log-in service 78 also generates the above-described list of services (step 1203). The ticket 82 and the list of services are then downloaded to the client 1 (step 1204).

3. Asynchronous Notification to Clients by Server

Another limitation associated with prior art Internet servers is the inability to provide asynchronous notification information to the client in the absence of a request from the client to do so. It would be desirable, for example, for a server to notify a client on its own initiative when a particular Web page has changed or that a particular service is inaccessible. The server 5 of the present invention provides such capability, and the client 1 is configured to receive and decode such notifications. For example, the client 1 can receive updates of its listing of service providers from the server 5 at various points in time, as already described. Similarly, if a particular service provider becomes unavailable, that fact will be automatically communicated to the client 1. As another example, if e-mail addressed to the user has been received by the server 5, then the server 5 will send a message to the client 1 indicating this fact. The client 1 will then notify the user that e-mail is waiting by a message displayed on the television set 12 or by an LED (light emitting diode) built into the housing of WebTV™ box 10.

Thus, a method and apparatus have been described for providing proxying and transcoding of documents in a network. The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

17

What is claimed and desired to be secured by United States Letters Patent is:

1. In a proxying server coupled to a client and to a remote server, the proxying server operating as a proxy on behalf of the client for accessing the remote server, a method of retrieving and transcoding a document requested by the client, the method comprising the steps of:

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document in order to perform at least one function selected from the following functions:

matching decompression requirements at the client; converting the document into a format compatible for the client;

reducing latency experienced by the client; and altering the document to fit into smaller memory space; and

transmitting the document to the client.

2. A method according to claim 1, wherein the request includes a URL (Uniform Resource Locator).

3. A method according to claim 1, wherein the step of transcoding is performed while the step of retrieving the document is performed.

4. A method according to claim 1, wherein the step of transcoding comprises the step of reading at least a portion of the document from a proxy cache located in the proxying server.

5. A method according to claim 1, wherein the step of transmitting the document to the client is performed prior to performing the step of transcoding, at the proxying server, the data in the document.

6. A method according to claim 1, wherein the step of transmitting the document to the client comprises the step of transmitting the transcoded document to the client, wherein the step of transmitting the document to the client is performed after the step of transcoding, at the proxying server, the data in the document.

7. A method according to claim 1, wherein the document includes a link to another document, the link including a retrieval address, and wherein the step of transcoding at the proxying server the data in the document comprises the step of updating the link

8. A method according to claim 7, wherein the other document is an image, and wherein the step of updating the link comprises the step of adding information to the document indicating the size of the image.

9. A method according to claim 8, wherein the other document is inaccessible to the proxying server, and wherein the step of updating the link comprises the step of removing the link.

10. A method according to claim 7, wherein the other document has been relocated from the retrieval address to a redirect address, and wherein the step of updating the link comprises the step of updating the link to correspond to the redirect address.

11. A method according to claim 1, wherein the client includes a television display, wherein the document references an image, and wherein the step of transcoding, at the proxying server, the data in the document is conducted in order to perform at least the function of converting the document into a compatible format for the client, and comprises the step of revising the data such that the image is sized for display on the television display.

18

12. A method according to claim 1, wherein the document references an image having a first image format, and wherein the step of transcoding, at the proxying server, the document is conducted in order to perform at least the function of converting the document into a compatible format for the client, and includes the step of converting the first image format to a second image format.

13. A method according to claim 1, further comprising the steps of:

identifying an image referenced by the document;

determining whether the image has been previously retrieved by the proxying server; and

if the image has been previously retrieved by the proxying server, accessing information stored in the proxying server indicating the size of the image;

wherein the step of transcoding, at the proxying server, the data in the document is conducted in order to perform at least the function of reducing latency experienced by the client, and comprises the step of using the information indicating the size of the image to revise the data of the document to allow the document to be partially displayed by the client before the image is received by the client.

14. A method according to claim 13, wherein the step of transcoding, at the proxying server, the data in the document comprises the following step:

if the image has been previously retrieved by the proxying server, transcoding at the proxying server the data in the document without the image but providing a space for the image to be later inserted; and

wherein the step of transmitting the document to the client comprises the following steps:

transmitting the document to the client without the image but providing the space for the image to be later inserted; and

after transmitting the document to the client without the image, transmitting the image to the client.

15. In a proxying server coupled to a client and to a remote server, the proxying server operating as a proxy on behalf of the client for accessing the remote server, a method of retrieving and transcoding a document requested by the client, the method comprising the steps of:

providing a persistent database at the proxying server, the persistent database including information relating to the document;

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document using the information included in the persistent database in order to perform at least one function selected from the following functions:

matching decompression requirements at the client; converting the document into a format compatible for the client;

reducing latency experienced by the client; and altering the document to fit into smaller memory space; and

transmitting the document to the client.

16. A method according to claim 15, further comprising the step of storing in the persistent database validity information corresponding to the document.

19

17. A method according to claim 16, wherein the step of retrieving the document comprises the following steps:

retrieving the document from the persistent database if the validity information corresponding to the document indicates that the document is valid; and

retrieving the document from the remote server if the validity information corresponding to the document indicates that the document is not valid.

18. A method according to claim 16, wherein the step of storing in the persistent database validity information corresponding to the document comprises the step of observing a rate of change of the document.

19. A method according to claim 18, wherein the step of observing a frequency of change of the document comprises the step of periodically retrieving the document, wherein successive retrievals of the document are separated in time by at least a time interval.

20. A method according to claim 19, wherein the step of periodically retrieving the document comprises the following steps:

retrieving the document at a first time;

storing the document retrieved at the first time in a memory;

retrieving the document at a second time, the second time being at least the time interval after the first time;

determining whether the document retrieved at the second time has changed compared to the document retrieved at the first time;

if the document retrieved at the second time is different than the document retrieved at the first time, replacing the document retrieved at the first time with the document retrieved at the second time in the memory; and

if the document retrieved at the second time is the same as the document retrieved at the first time, extending the time interval.

21. A method according to claim 15, further comprising the step of storing in the persistent database performance information relating to performance of the remote server when accessing the document.

22. A method according to claim 21, wherein the performance information comprises a latency value.

23. A method according to claim 15, further comprising the step of storing in the persistent database information for optimizing memory usage by the client.

24. A method according to claim 15, wherein the step of retrieving the document comprises the following steps:

determining whether a redirect address corresponding to the document is stored in the persistent database;

if the redirect address corresponding to the document is stored in the persistent database, accessing the remote server using the redirect address, the remote server corresponding to the redirect address; and

if the redirect address corresponding to the document is not stored in the persistent database, the method further comprises the following steps:

accessing a previous remote server at which the document previously resided;

obtaining the redirect address from the previous remote server;

storing the redirect address in the persistent database; and

20

accessing the remote server using the redirect address.

25. A computer program product for implementing, in a proxying server coupled to a client and to a remote server, a method of retrieving and transcoding a document requested by the client, the computer program product comprising a computer-readable medium carrying computer-executable instructions for causing the proxying server to perform acts included in the method, said acts comprising:

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document in order to perform at least one function selected from the following functions:

matching decompression requirements at the client;

converting the document into a format compatible for the client;

reducing latency experienced by the client; and

altering the document to fit into smaller memory space; and

transmitting the document to the client.

26. A computer program product according to claim 25, wherein the computer-executable instructions comprise a plurality of program code means for transcoding at least a portion of the document, including:

program code means for transcoding at least a portion of the document so as to match decompression requirements at the client;

program code means for transcoding at least a portion of the document so as to convert the document into a format compatible with the client;

program code means for transcoding at least a portion of the document so as to reduce latency experienced by the client; and

program code means for transcoding at least a portion of the document so as to alter the document to fit into smaller memory space.

27. A computer program product according to claim 26, wherein the act of transcoding comprises selecting at least one of said plurality of program code means for transcoding for use in the act of transcoding.

28. A computer program product according to claim 25, wherein the act of transcoding is performed while the act of retrieving the document is performed.

29. A computer program product according to claim 25, wherein the act of transcoding comprises reading at least a portion of the document from a proxy cache located in the proxying server.

30. A computer program product according to claim 25, wherein the act of transmitting the document to the client is performed prior to performing the act of transcoding.

31. A computer program product according to claim 25, wherein the act of transmitting the document to the client comprises the act of transmitting the transcoded document to the client, wherein the act of transmitting the document to the client is performed after the act of transcoding.

* * * * *



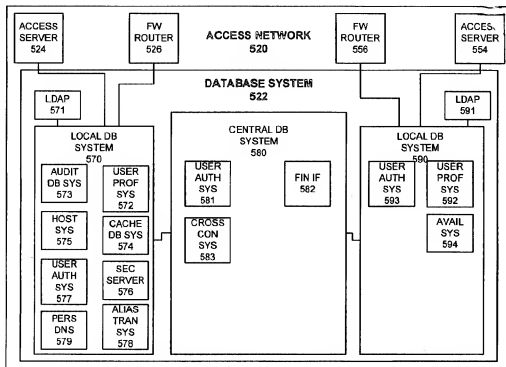
US006697806B1

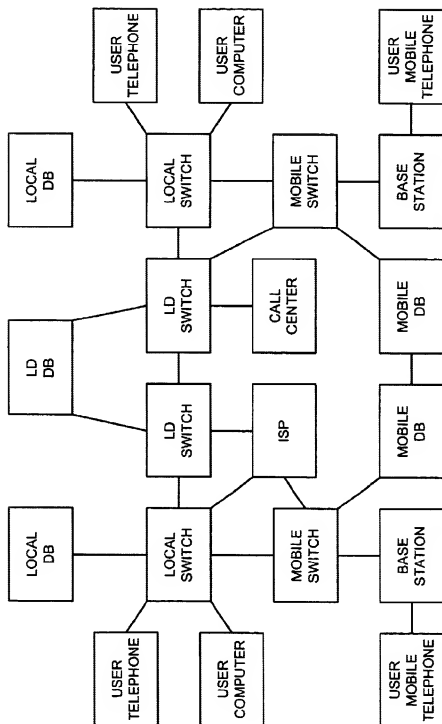
**(12) United States Patent
Cook****(10) Patent No.: US 6,697,806 B1
(45) Date of Patent: Feb. 24, 2004****(54) ACCESS NETWORK AUTHORIZATION****(75) Inventor: Fred S. Cook, Olathe, KS (US)****(73) Assignee: Sprint Communications Company,
L.P., Overland Park, KS (US)****(*) Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.**(21) Appl. No.: 09/574,978****(22) Filed: May 19, 2000****Related U.S. Application Data****(63)** Continuation of application No. 09/556,276, filed on Apr. 24, 2000.**(51) Int. Cl.⁷ G06F 17/30****(52) U.S. Cl. 707/10; 707/3; 707/9;
709/227; 709/203; 709/219****(58) Field of Search 707/1-3, 9-10,
707/4, 104.1; 709/223-227, 219, 203****(56) References Cited****U.S. PATENT DOCUMENTS**5,884,312 A * 3/1999 Dustan et al. 707/10
6,151,601 A * 11/2000 Papierniak et al. 707/1

* cited by examiner

Primary Examiner—Jean R. Homere
Assistant Examiner—Mohammad Ali**(57) ABSTRACT**

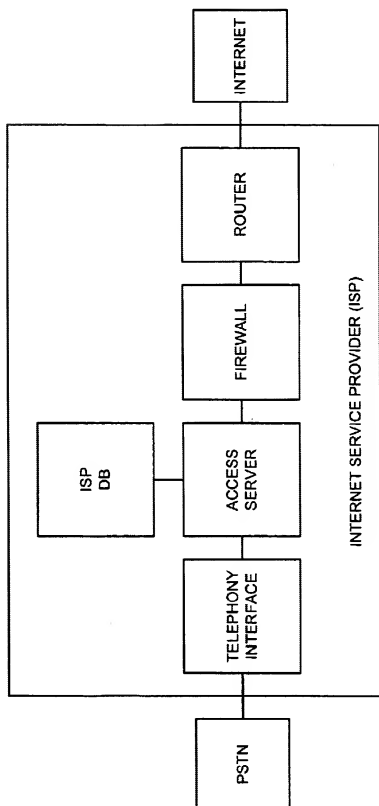
An access communication system provides access between a user system and a plurality of communication networks. The plurality of communication networks provide services to a user in the user system. An access communication system includes a local database system and an access server that is connected to the user system and the plurality of communication networks. The local database system receives a user logon. The local database system then processes the user logon to determine if the user is allowed access to the access communication system based on a local database system. The local database system then provides access to the access communication system to the user in response to the determination that the user is allowed access based on the local database system. The local database system then generates an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system. The local database system receives and processes an authorization response indicating whether the user is allowed to use the access system from the second database system. The local database system then provides access to the access communication system to the user in response to the authorization response that allows the user to use the access communication system.

24 Claims, 63 Drawing Sheets



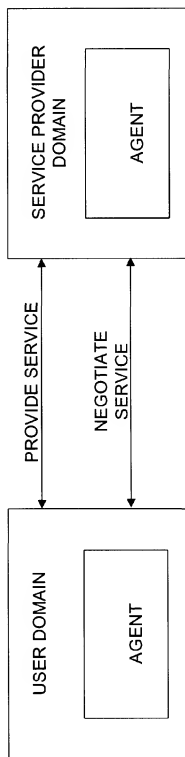
PRIOR ART

FIG. 1



PRIOR ART

FIG. 3



PRIOR ART

FIG. 4

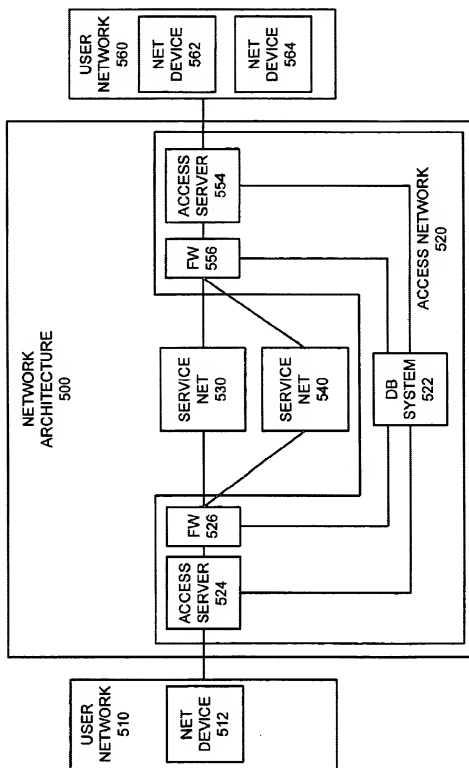


FIG. 5

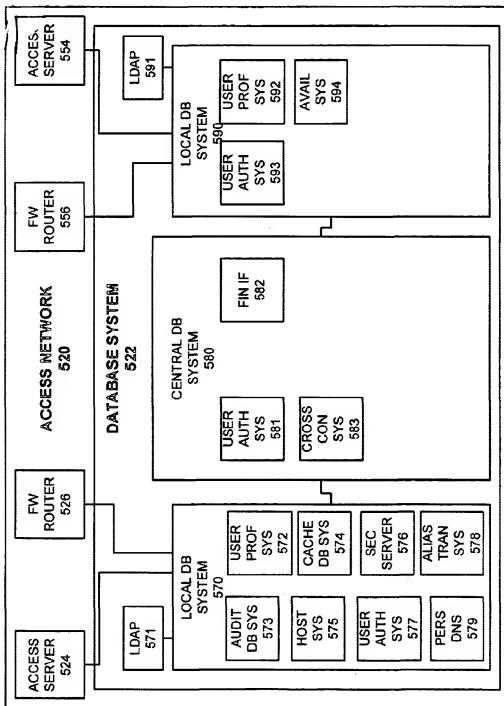


FIG. 6

USER ID	PASSWORD	NAME	ACCOUNT NUMBER	SERVICES	ADDRESS	BILLING CODE	CLASS	GROUP	SHELL	MACROS

FIG. 7

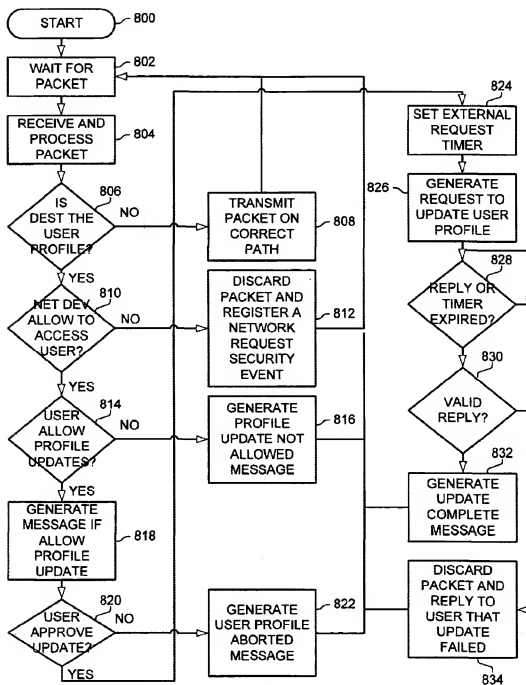


FIG. 8

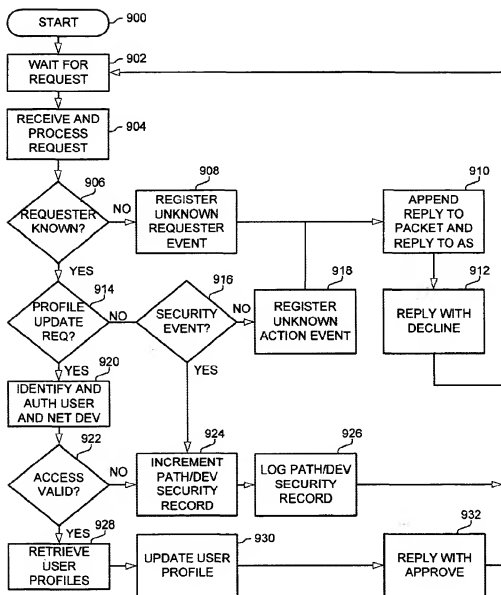


FIG. 9

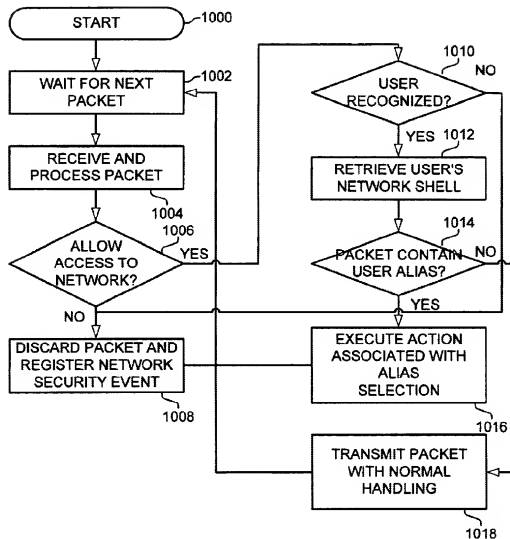


FIG. 10

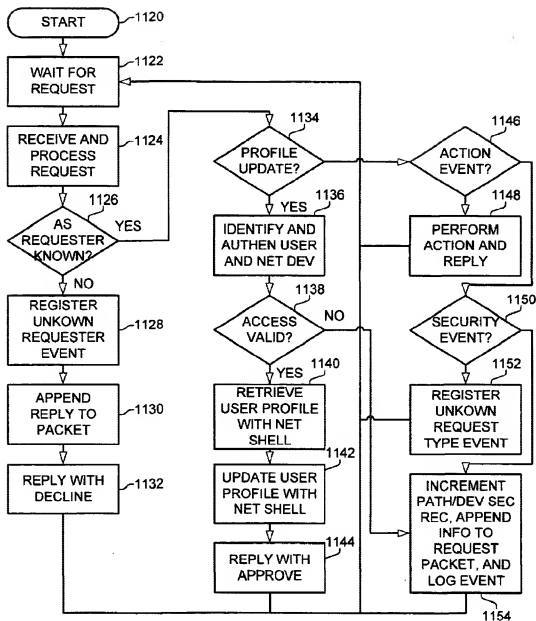


FIG. 11

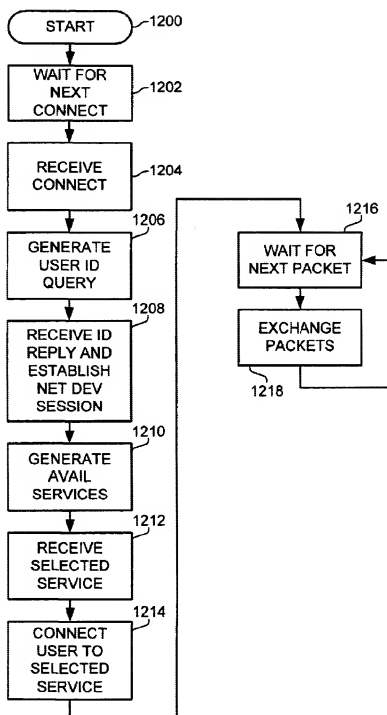


FIG. 12

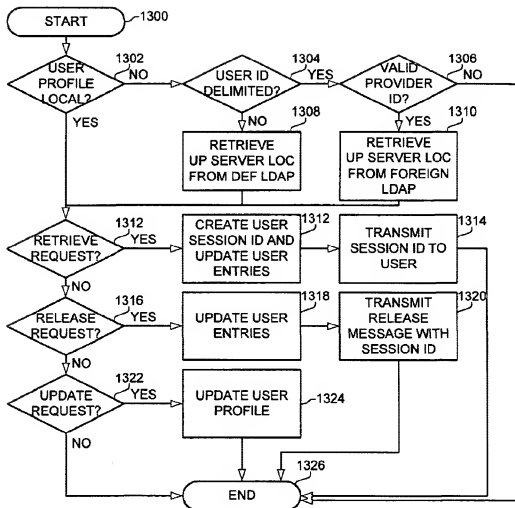


FIG. 13

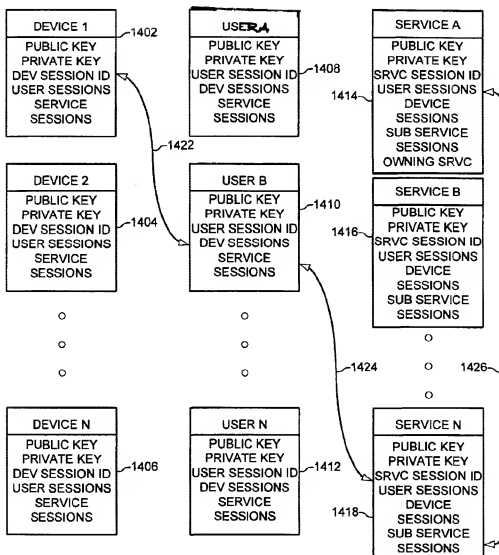
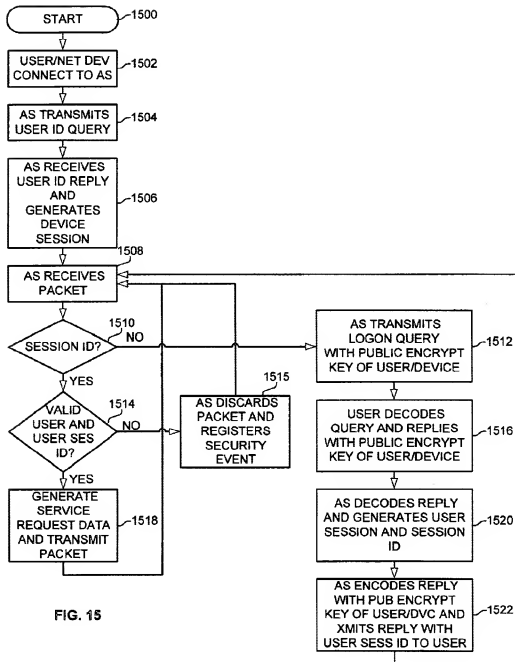


FIG. 14



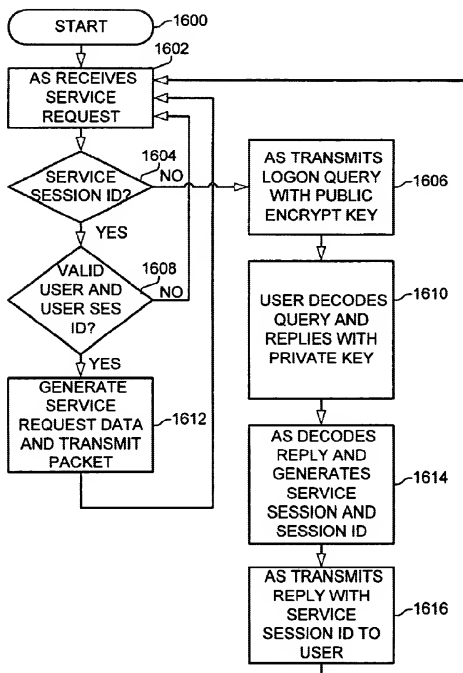


FIG. 16

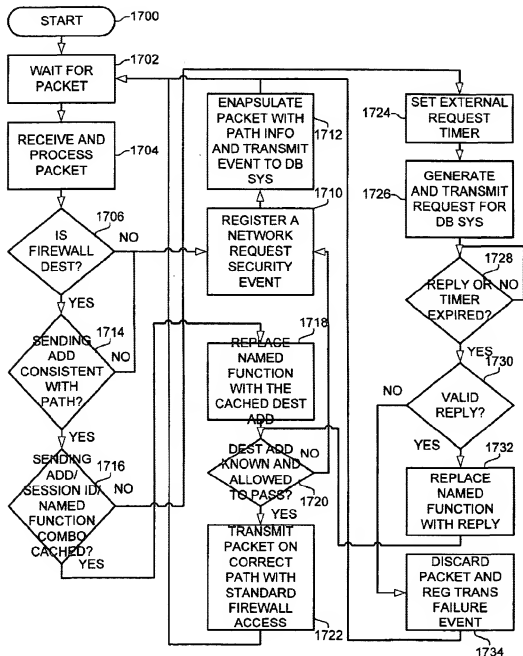


FIG. 17

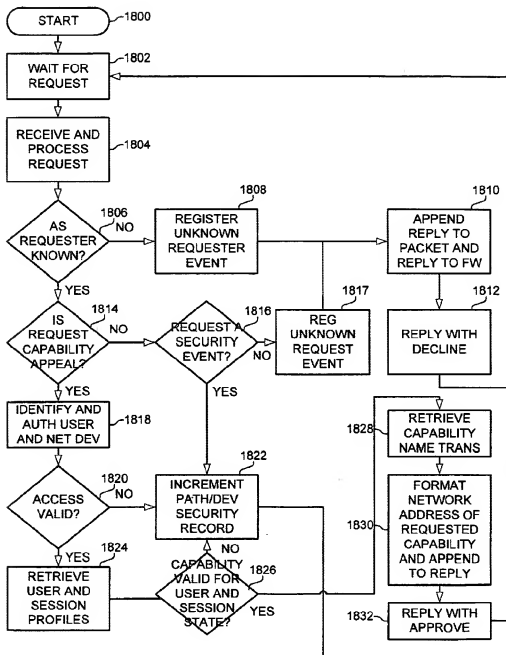


FIG. 18

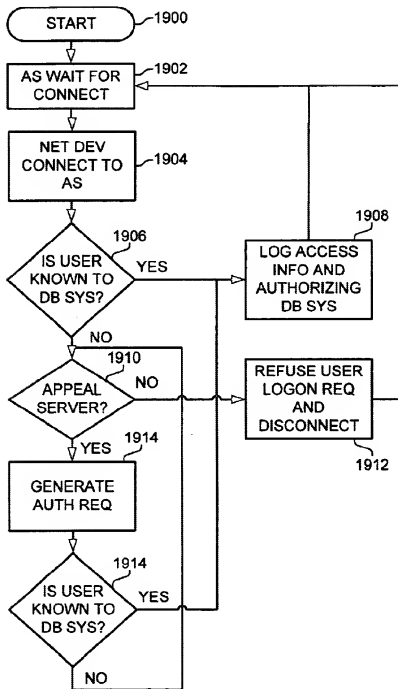


FIG. 19

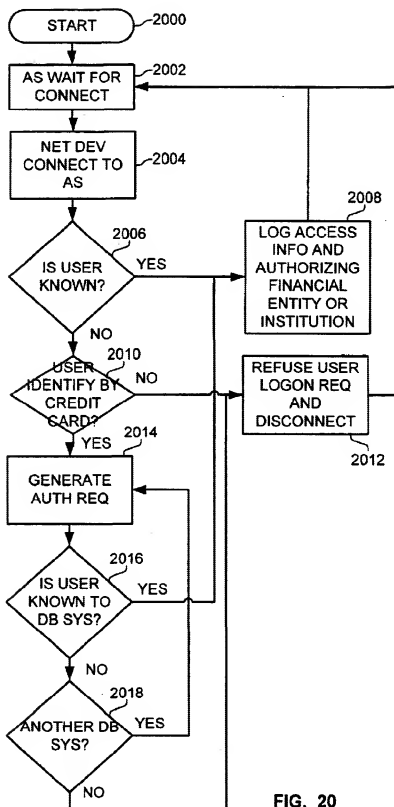


FIG. 20

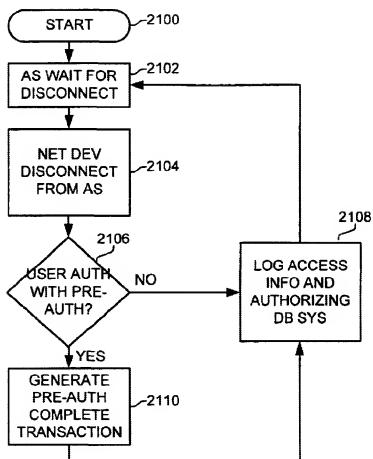


FIG. 21

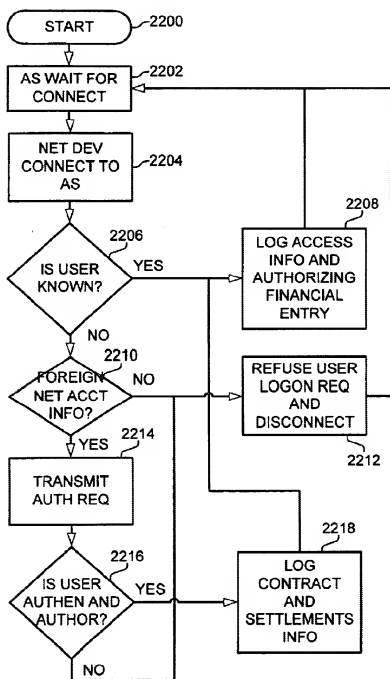


FIG. 22

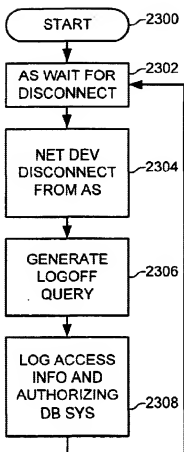


FIG. 23

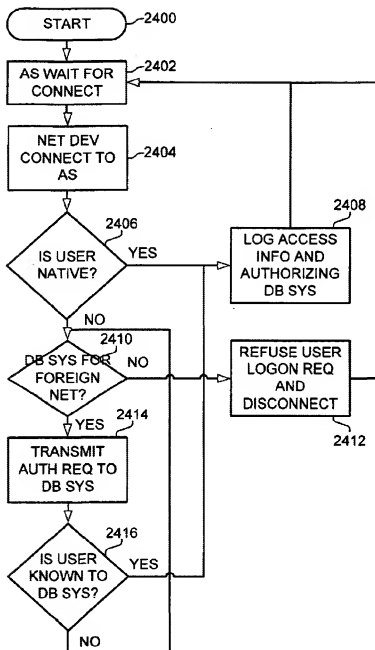


FIG. 24

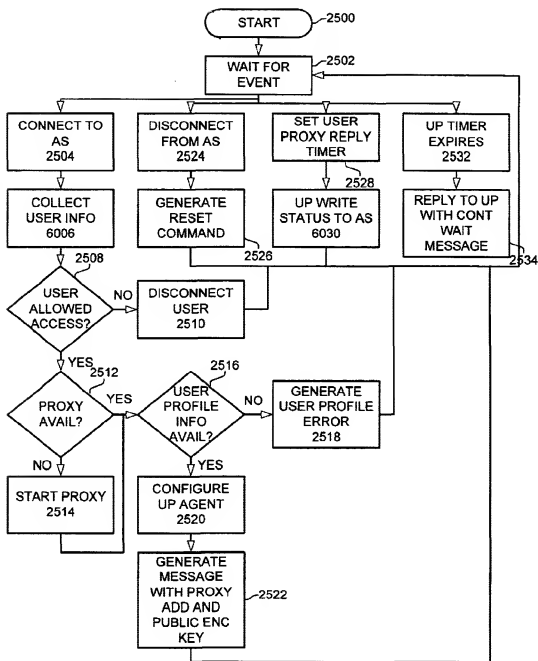


FIG. 25

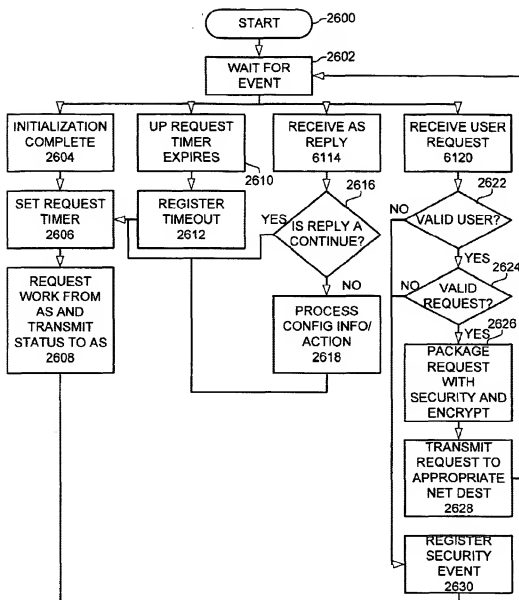


FIG. 26

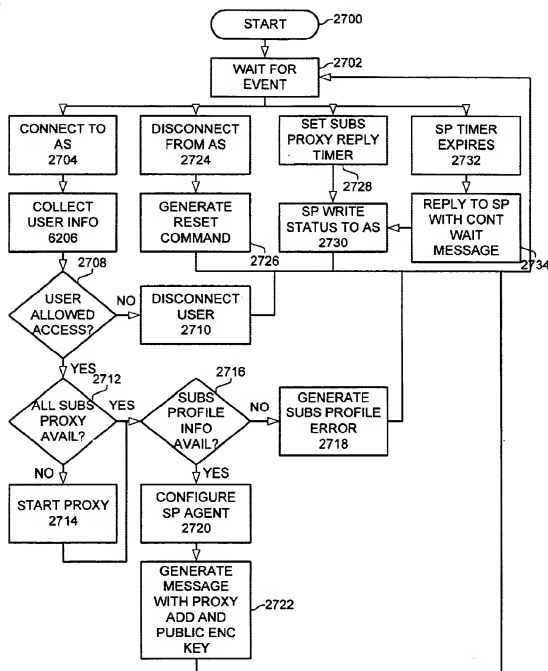


FIG. 27

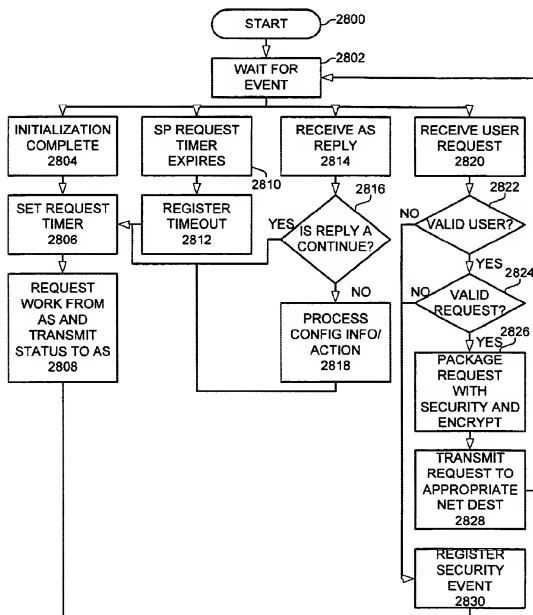


FIG. 28

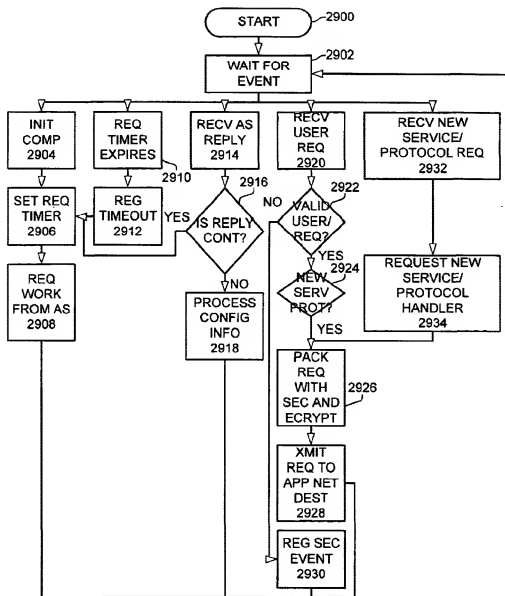


FIG. 29

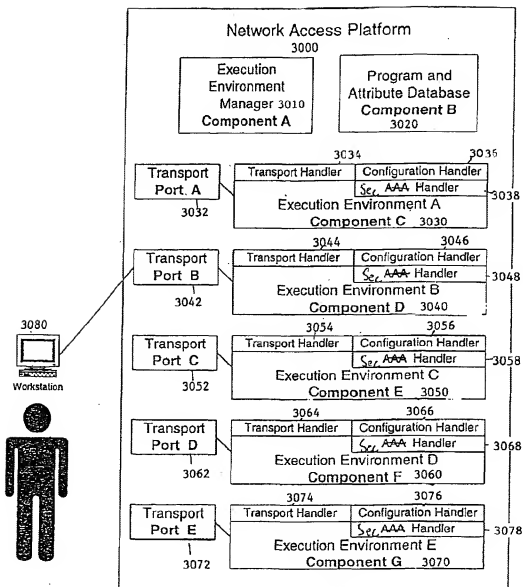


FIGURE 30

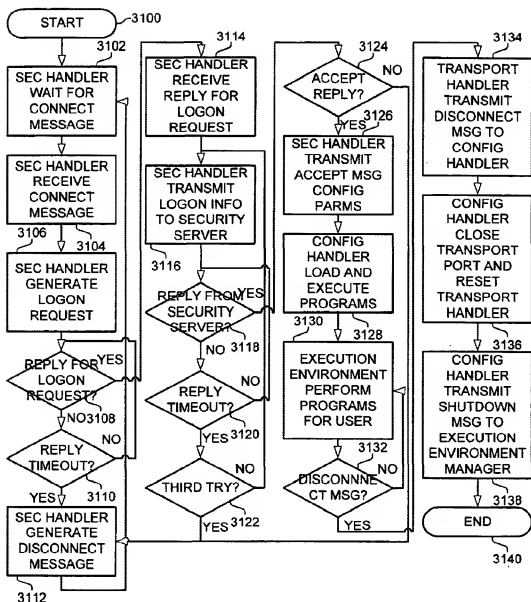


FIG. 31

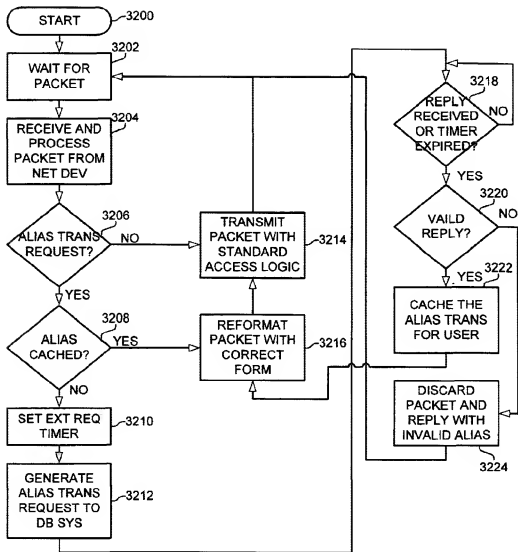


FIG. 32

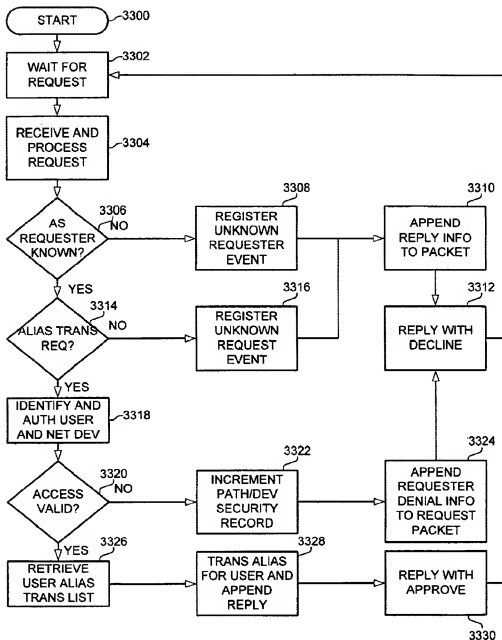


FIG. 33

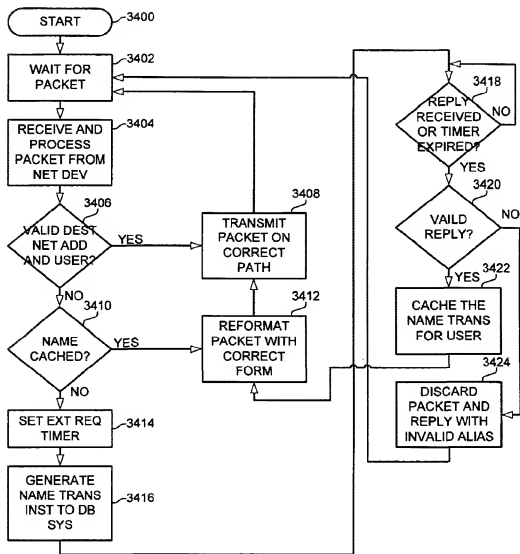


FIG. 34

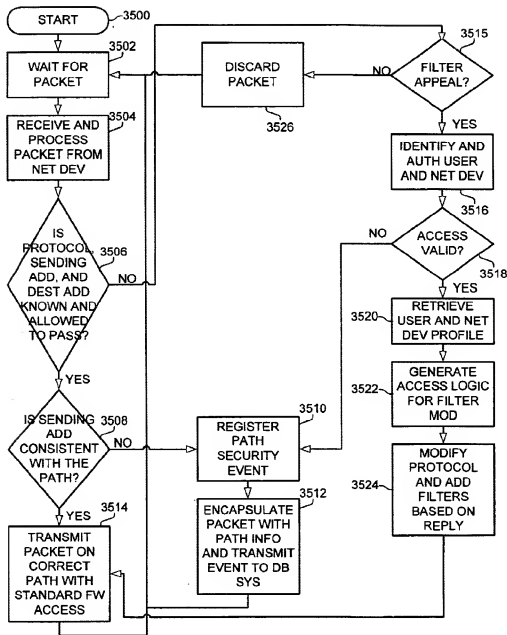


FIG. 35

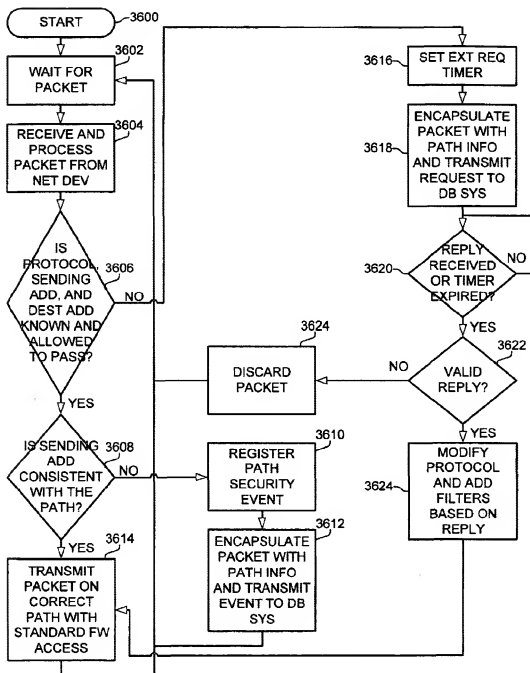


FIG. 36

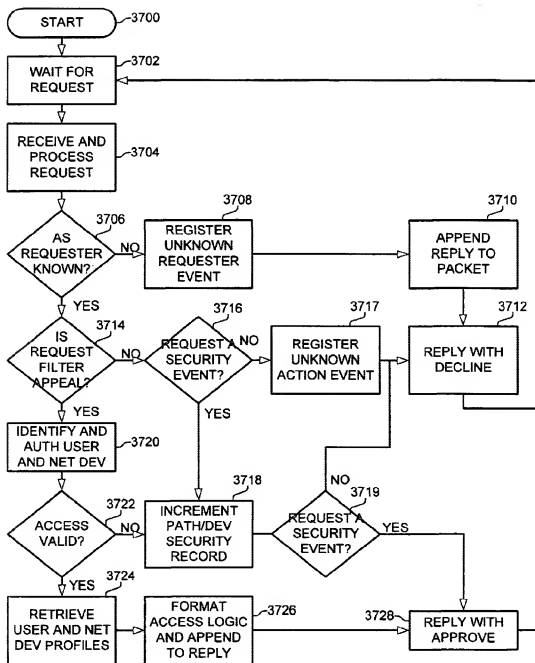


FIG. 37

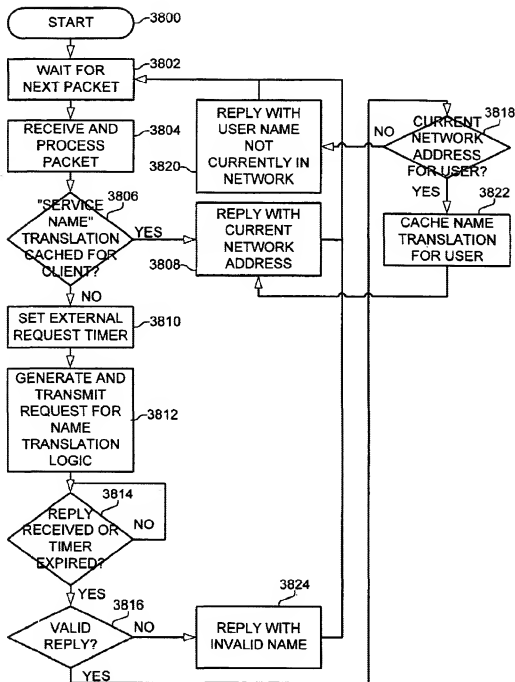


FIG. 38

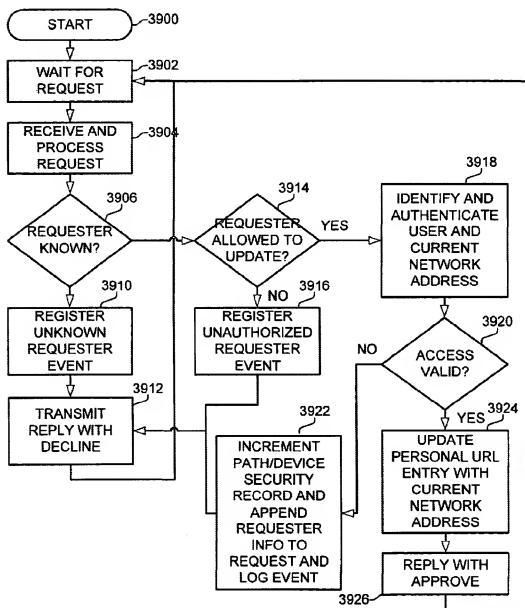


FIG. 39

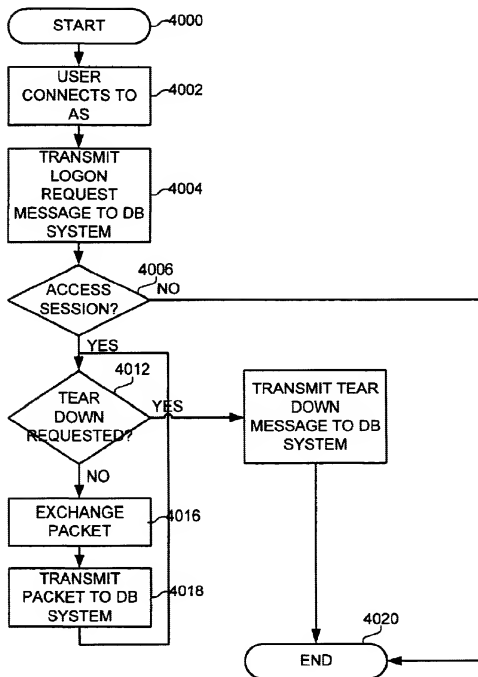


FIG. 40

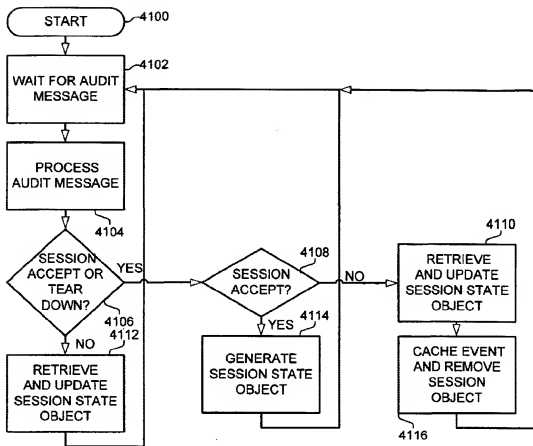


FIG. 41

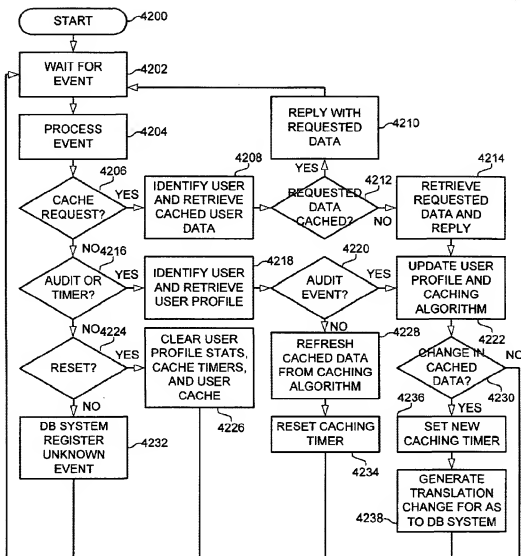


FIG. 42

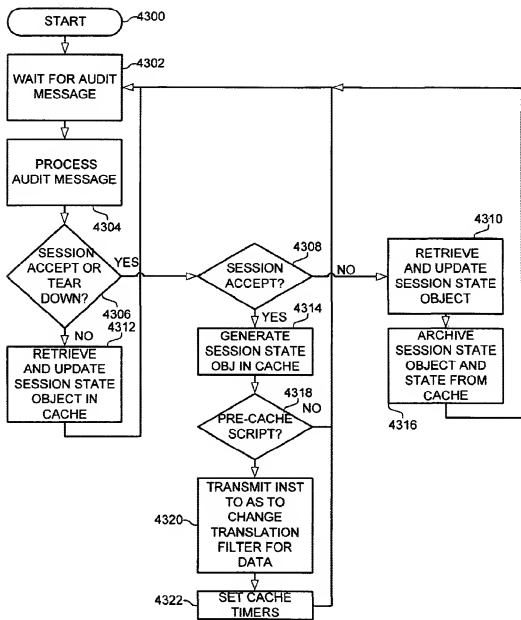


FIG. 43

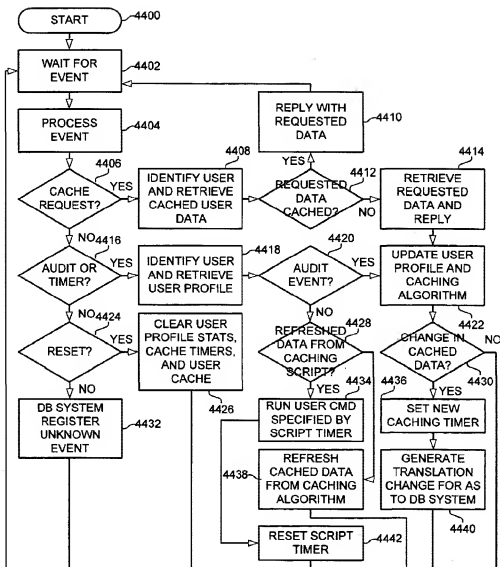


FIG. 44

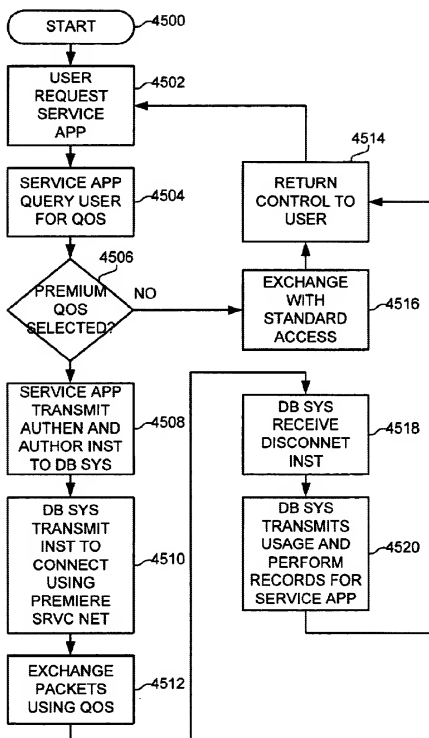


FIG. 45

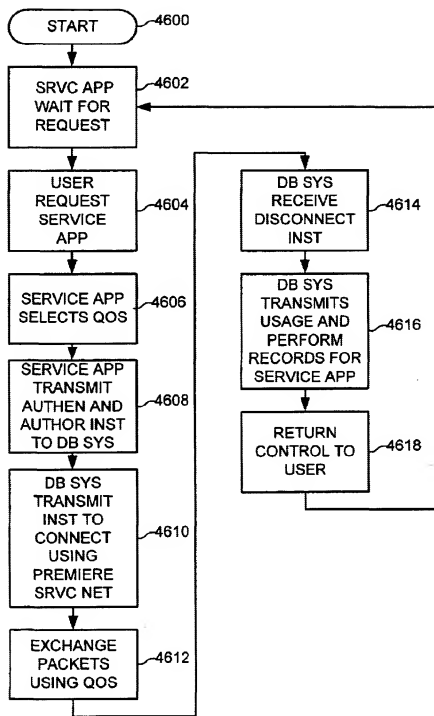


FIG. 46

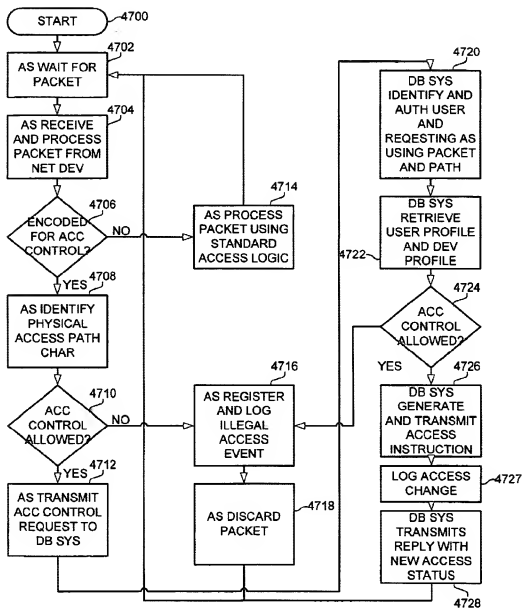


FIG. 47

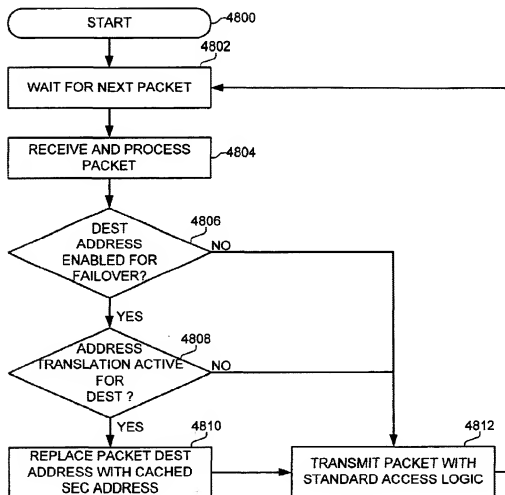


FIG. 48

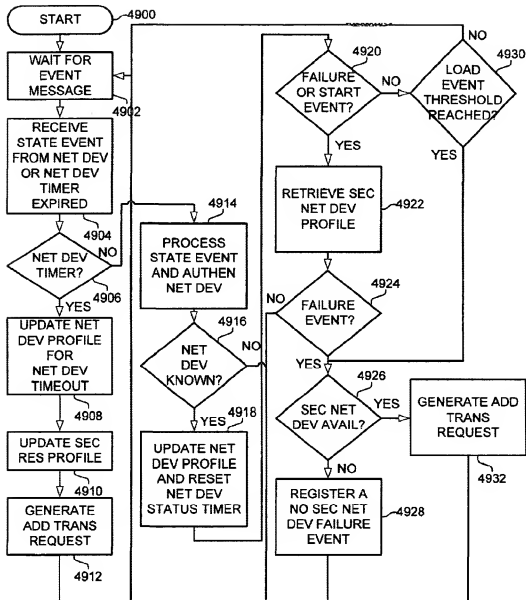


FIG. 49

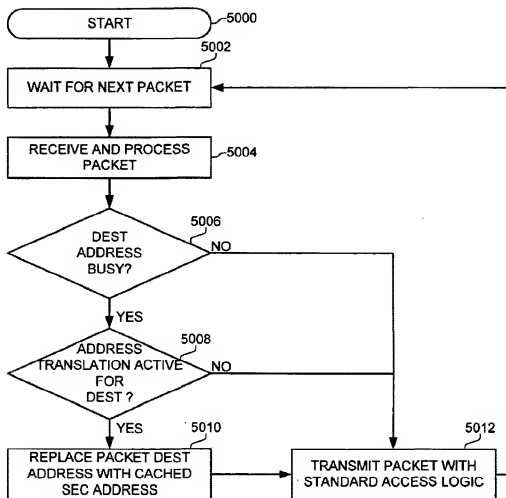


FIG. 50

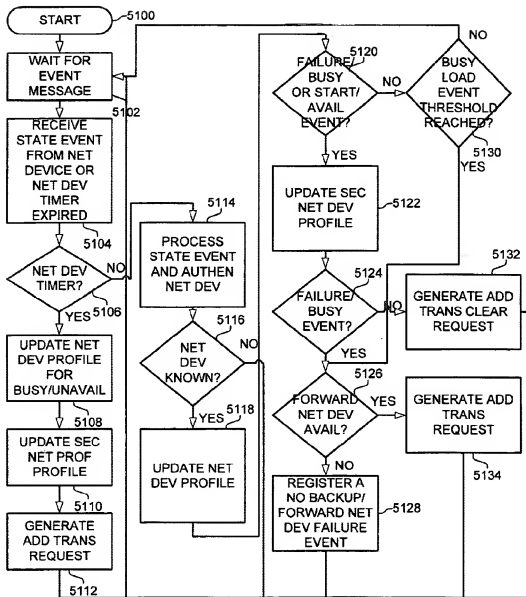


FIG. 51

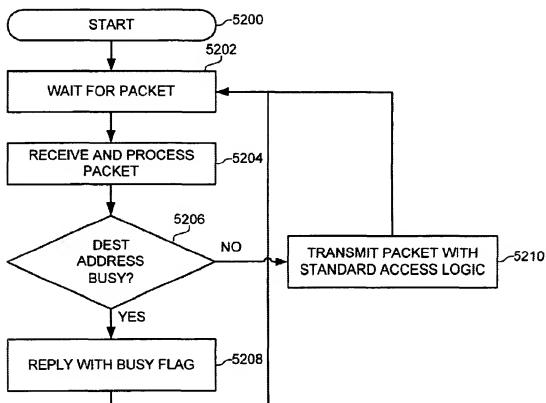


FIG. 52

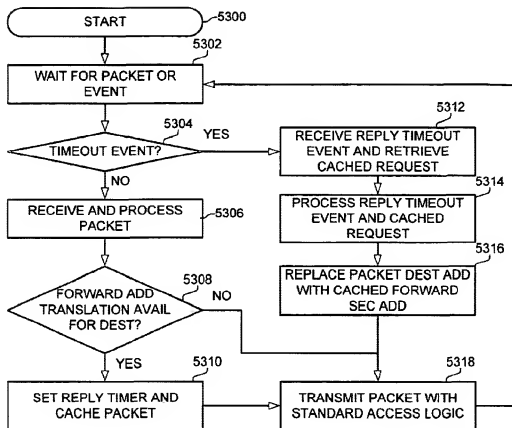


FIG. 53

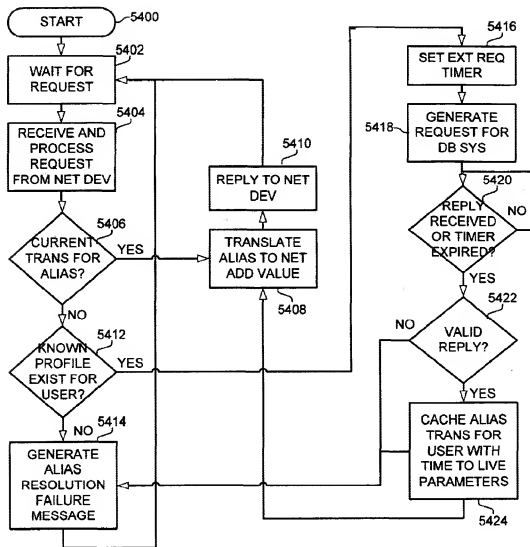


FIG. 54

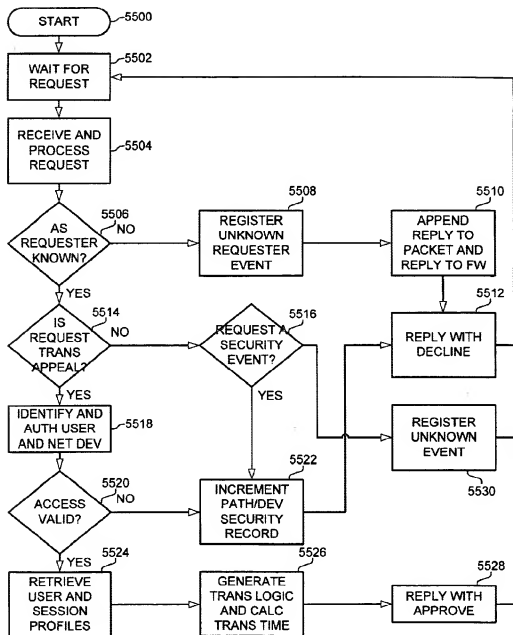


FIG. 55

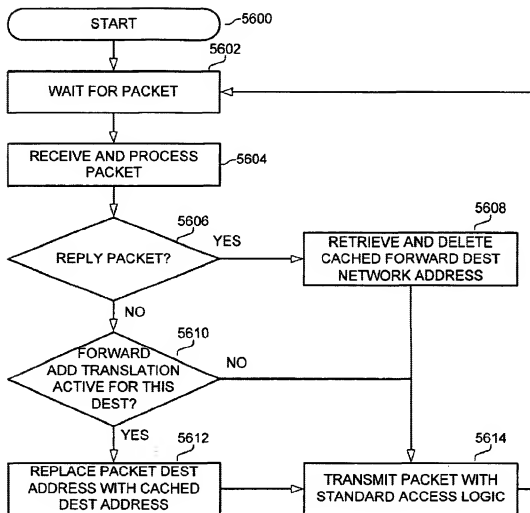


FIG. 56

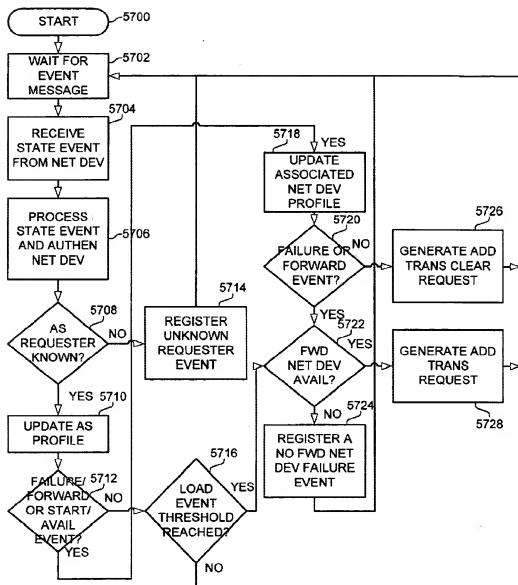


FIG. 57

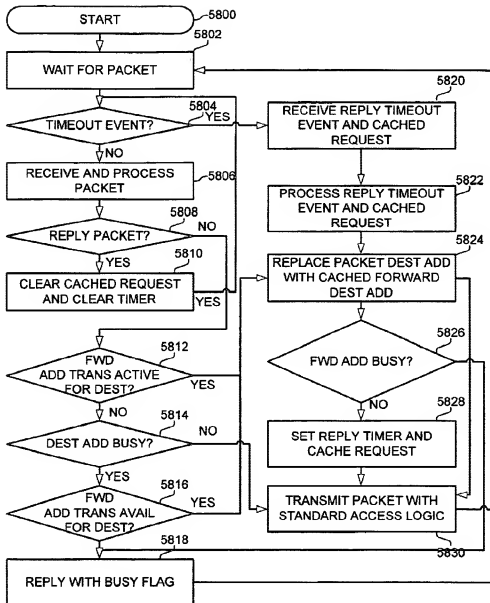


FIG. 58

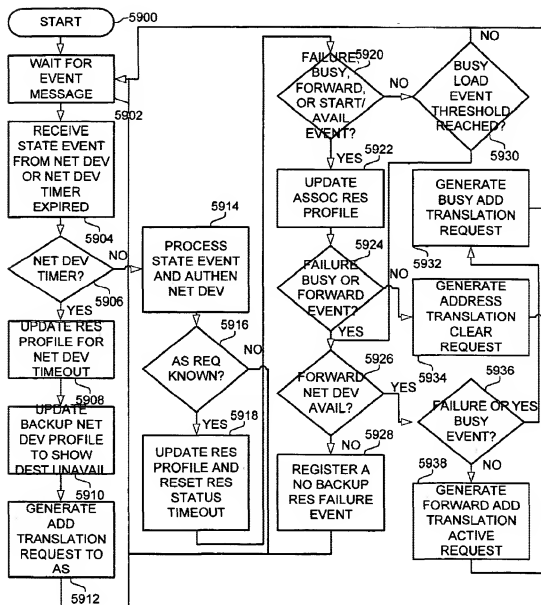
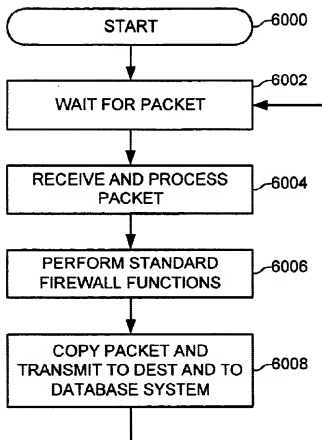


FIG. 59

**FIG. 60**

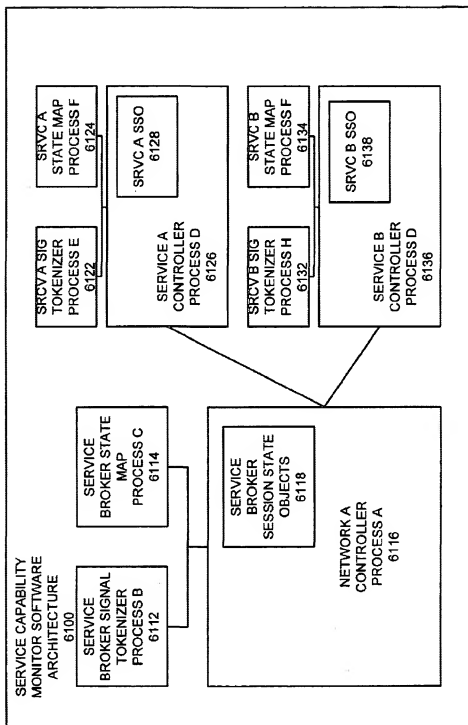
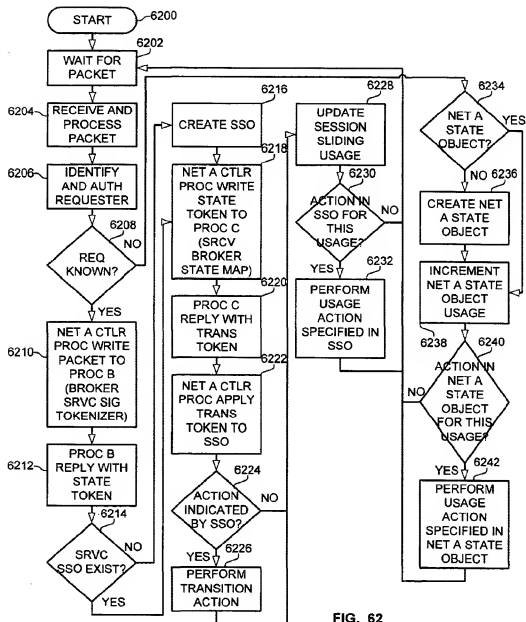


FIG. 61



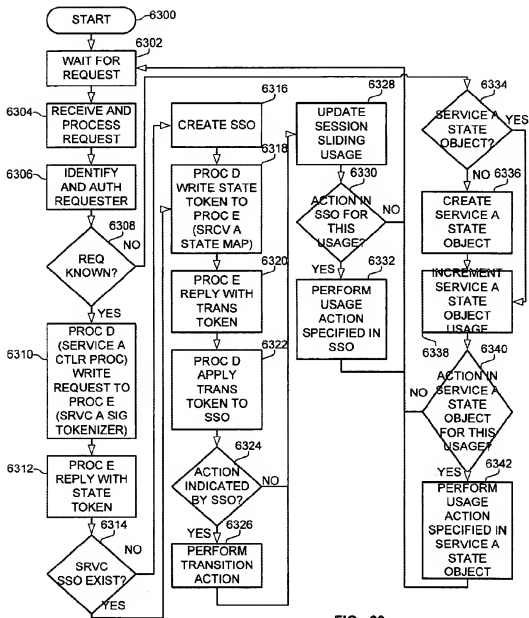


FIG. 63

ACCESS NETWORK AUTHORIZATION

RELATED APPLICATIONS

This application is a continuation of prior application Ser. No. 09/556,276, entitled "Access Communication System", filed Apr. 24, 2000, currently pending, and incorporated by reference into this application.

FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

Not applicable

MICROFICHE APPENDIX

Not applicable

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention is related to the field of communication networks, and in particular, to an access communication system that provides access to multiple service provider systems. More particularly, this invention relates to a system that authorizes access to use the access communication system.

2. Description of the Prior Art

Communication networks have seen dramatic development over the past several years so that today, there are multiple diverse communication networks providing services. The current technical challenge is to develop interfaces between the networks to provide seamless service across multiple networks. Unfortunately, today's interfaces lack the ability to offer the user with easy access to services from multiple systems. These interfaces do not customize operations for the user.

FIG. 1 illustrates the conventional public telephone network. User telephones and computers are connected to local switches. The local switches are coupled to a local database. The user places calls through the local switch. The local switch processes the called number to provide an end-point connection or access to other networks. The local switch may connect the call to another telephone in the local calling area. In a Local Number Portability (LNP) situation, the local switch exchanges information with the local database to obtain the appropriate routing number for a ported call. The local switch may also connect the call to an Internet Service Provider. If a Digital Subscriber Line (DSL) is used, DSL equipment may be used to bypass the local switch. The local switch may also connect the call to a long distance switch. To provide access to the long distance switch, the local switch first exchanges information with the local database. The local database identifies the long distance network for either the user or the dialed number.

The long distance switch processes the called number to route the call to another system or network. Prior to routing, the long distance switch validates the call by checking the caller's number. The long distance switch may also exchange information with the long distance database to provide special call-handling. One example is a calling card call, where the long distance database validates an account number for billing the call. Another example is a toll-free call, where the long distance database processes external information customized by the called party to route the call. Toll-free routing information includes items such as time and date, caller location, and call center status. Many long distance calls are simply routed through the long distance

network to another local network for call completion. Other calls are routed from the long distance network to a call center. Call centers offer a concentration of call-handling capabilities for operations, such as order entry, customer service, and promotions. Call centers include automatic call distribution equipment to route calls to the appropriate destination within the call center.

Both local and long distance networks exchange calls with mobile switches. The mobile switches are connected to base stations that communicate over the air with wireless telephones. When a mobile user places a call, the mobile switches exchange information with a mobile database to validate the mobile caller. The call is then routed to another mobile caller, the telephone network, or to an ISP. When a mobile caller moves around, their wireless phone logs-in with the physically proximate mobile network, and that mobile network updates the mobile user's location in the mobile databases. When a call is placed to that mobile user, their home mobile switch obtains a routing number from the mobile databases to route the call to the mobile network currently in communication with the mobile user.

FIG. 2 illustrates a conventional data network that transfers packets of user data to a destination based on address information carried in the packets. Users are connected to Local Area Networks (LANs) that are connected to Wide Area Networks (WANs). A common LAN is an Ethernet system. A common WAN is an intranet. WANs are interconnected by data networks, such as IP, TI, frame relay, or Asynchronous Transfer Mode (ATM). WANs are connected to the Internet through ISPs. WANs are connected to the public telephone network through telephony gateways. A common telephony gateway is a Private Branch Exchange (PBX).

FIG. 3 illustrates a conventional ISP. The public telephone network is coupled to a telephony interface that converts between telephony analog and digital protocols and the Internet Protocol (IP). Some telephony interfaces also handle DSL traffic that may already use IP. The telephony interface transfers IP traffic through an access server and firewall to a router. Some ISPs combine the firewall and the access server into one system. Also, the position of the firewall may vary, and traffic shapers may be present. The router exchanges IP traffic with the Internet.

In operation, the user calls the ISP over the telephone network and logs-in at the access server. The access server collects and forwards the user name and password to the ISP database. The ISP database validates the user name and password and returns an IP address to the access server. The IP address is for the user's terminal connection. Using the IP address, the user may communicate through the firewall to the router for transmissions to an IP address. The user now has Internet access through the router and exchanges packets with various Internet servers.

IP addresses are referred to as network addresses and include a network ID and a host ID. Network IDs are unique across the Internet and host IDs are unique within a given network. IP addresses are lengthy numerical codes, so to simplify things for the user, service addresses are available that are easier to remember. The service addresses are often the name of the business followed by ".com". Domain Name Service (DNS) is hosted by servers on the Internet and translate between service addresses and network addresses. The browser in the user computer accesses the DNS to obtain the desired network address.

FIG. 4 illustrates conventional network access. A current proposal for communication network access is provided by

3

the Telecommunication Information Network Architecture Consortium (TINA-C). TINA-C proposes the use of agents in the user domain and the service provider domain. The service provider domain could be a telephone network, data network, or ISP. The agents negotiate access service rights. Once the service is negotiated, the user receives the service from the service provider network during a service session. Unfortunately, the access session occurs between the user domain and a particular service provider domain. At present, the service provider domain provides limited access capability beyond simply handing off communications to another network based on a called number or network address. As a result, the ability to customize services for a particular user across multiple service providers is inadequate.

SUMMARY OF THE INVENTION

The inventions solve the above problems by providing access between a user system and a plurality of communication networks. The plurality of communication networks provide services to a user in the user system. An access communication system includes a database system and an access server that is connected to the user system and the plurality of communication networks.

In one aspect of the inventions for user access profile inheritance, the database system receives an update request from the access server to update a user access profile through inheritance. The database system then processes the update request to inherit user profile information from a user profile data structure. The database system updates the user access profile with the user profile information.

In another aspect of the inventions for network shells, the access server receives an alias selection from a user for a network shell that includes alias selections associated with actions. The access server then processes the alias selection to execute an action associated with the alias selection.

In another aspect of the inventions for service based directory, the access server transmits a list of services to a user system. The access server then receives a selection from the list of services. The access server processes the selection to generate an instruction to provide the service related to the selection.

In another aspect of the inventions for user access profile mobility, the database system receives user information. The database system then processes the user information to determine if a user access profile is local within a local database system. The database system generates and transmits a request to retrieve a user access profile from a second database system external to the local database system in response to the determination that the user access profile is not local.

In another aspect of the inventions for service, user, and device sessions, the access communication system establishes a connection between a network device and the access server. The access communication system then generates a device session including a device session ID based on the network device. The access communication system generates and transmits a login query for the network device. The access communication system receives and processes a login reply from the network device to generate a user session including a user session ID based on the user. The access communication system receives and processes a request for the service to generate a service session including a service session ID based on the service. The service may generate and transmit a login query for the user. The access communication system links the device session, user session, and the service session using the device session ID, the user session ID, and the service session ID.

4

In another aspect of the inventions for service capability firewall, the access server receives information including a named function request for a service provider. The access server processes the information to check if the named function request is valid for the service provider and the service. If valid, the access server determines if a private destination address exists for the named function request. The access server replaces the named function request with the private destination address in response to the determination that the private destination address exists for the named function request. The access server then transmits the information with the private destination address to the service provider.

In another aspect of the inventions for prepaid access and bank card access, the database system receives information identifying a billing code for a user. The database system then processes the billing code to determine if the user is allowed to use the access system. The database system provides access to the access system in response to the determination that the user is allowed to use the access system.

In another aspect of the inventions for global authentication and access card, the database system receives a user login. The database system then processes the user login to determine if the user is allowed access to the access communication system based on a local database system. The database system then provides access to the access communication system to the user in response to the determination that the user is allowed access based on the local database system. The database system then generates an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system. The database system receives and processes an authorization response indicating whether the user is allowed to use the access system from the second database system. The database system then provides access to the access communication system to the user in response to the authorization response that allows the user to use the access communication system.

In another aspect of the inventions for user based proxy and subscriber based proxy, the database system includes a user proxy. The user proxy receives a request for the service from the user system. The user proxy then transmits the request for the service to a service provider. The user proxy exchanges user information between the user system and the service provider.

In another aspect of the inventions for dynamic proxy, the database system includes a proxy. The proxy receives a service/protocol request for a new service or protocol. The proxy processes the service/protocol request to generate a handler request to obtain a handler for the new service or protocol. The proxy then receives and executes the handler for the new service or protocol.

In another aspect of the inventions for access execution environment, the database system receives and processes a login reply into an access execution environment for a user. The database system retrieves programs for the user into the access execution environment. The database system executes the programs for the user in the access execution environment.

In another aspect of the inventions for domain name scoping and inband domain name service lookup, the access server receives information including an alias from the user system. The access server determines if the alias exists in a cache including aliases and alias translations for the user.

5

The access server changes the information based on the cached alias translation.

In another aspect of the inventions for inline access service triggering, the access server receives information. The access server then processes the information to determine if the information is allowed to pass. The access server changes access logic based on the information in response to the determination that the information is not allowed to pass. The access server changes the filters of the access server based on the information in response to the determination that the information is not allowed to pass.

In another aspect of the inventions for access service triggering, the access server receives information. The access server processes the information to determine if the information is allowed to pass. The access server then generates a request from a database system in response to the determination that the information is not allowed to pass. The access server receives a reply including access logic from the database system. The access server changes filters of the access server based on the access logic.

In another aspect of the inventions for personal URL, the database system receives information including a user alias. The database system processes the information to determine if a user alias translation including a current network address for the user alias exists. The database system then modifies the information with the current network address using the user alias translation.

In another aspect of the inventions for predictive caching, the access server receives a request for data. The access server then determines if the data exists in a user cache wherein the user cache contains cached data based on the user's predictive patterns. The access server retrieves the data from the user cache in response to the determination that the data exists in the user cache. The access server transmits the request for data for the service provider in response to the determination that the data does not exist in the user cache.

In another aspect of the inventions for user controlled caching, the access server receives a request for data. The access server determines if the data exists in a user cache wherein the user cache contains cached data based on a user's script of commands. The access server retrieves the data from the user cache in response to the determination that the data exists in the user cache. The access server then transmits the request for data for the service provider in response to the determination that the data does not exist in the user cache.

In another aspect of the inventions for service usage audit, the access server receives an audit message into a database system. The access server processes the audit message to store the audit message in the database system.

In another aspect of the inventions for switching access by a user, switching access by a service provider, and dynamic access control, the database system receives a request. The database system processes the request to determine if the switching of the access is allowed. The database system then generates an instruction to switch access in response to the determination that the switching is allowed.

In another aspect of the inventions for network failover, network busy forwarding, time-out, busy flag, forwarding, and network endpoint availability management, the access server receives information for a destination network device. The access server determines if the destination network device is available. The access server performs an action in response to the determination that the destination network device is unavailable.

6

In another aspect of the inventions for scheduled alias translation, the access server receives information including an alias. The access server processes the information to determine whether an alias translation exists based on an alias translation schedule. The access server then modifies the information based on the alias translation in response to the determination the alias translation exists.

In another aspect of the inventions for service capability monitor, the database system receives information from an access server during a service session. The database system determines a current state of the service session based on the information. The database system determines a state transition based on the current state and a map of state transitions of the service. The database system determines whether the state transition is valid for the service session.

BRIEF DESCRIPTION OF THE DRAWINGS

The same reference number represents the same element on all drawings.

FIG. 1 illustrates a public telephone network in the prior art.

FIG. 2 illustrates a data network in the prior art.

FIG. 3 illustrates an Internet Service Provider in the prior art.

FIG. 4 illustrates conventional network access.

FIG. 5 illustrates a network architecture in an example of the invention.

FIG. 6 illustrates an access network in an example of the invention.

FIG. 7 illustrates a table for a user access profile in an example of the invention.

FIG. 8 illustrates a flowchart for an access server for inheriting a user access profile in an example of the invention.

FIG. 9 illustrates a flowchart for a database system for inheriting a user access profile in an example of the invention.

FIG. 10 illustrates a flowchart of an access server for executing a network shell in an example of the invention.

FIG. 11 illustrates a flowchart of a database system for updating a network shell in an example of the invention.

FIG. 12 illustrates a flowchart for the services based directory in an example of the invention.

FIG. 13 illustrates a flowchart of user access profile mobility in an example of the invention.

FIG. 14 illustrates a logical view of device, user, and service sessions in an example of the invention.

FIG. 15 illustrates a flowchart for a user based session in an example of the invention.

FIG. 16 illustrates a flowchart for a service based session in an example of the invention.

FIG. 17 illustrates a flowchart for a firewall/router for service capability firewall in an example of the invention.

FIG. 18 illustrates a flowchart for a database system for service capability firewall in an example of the invention.

FIG. 19 illustrates a flowchart for prepaid access in an example of the invention.

FIG. 20 illustrates a flowchart for bank card access for a connection in an example of the invention.

FIG. 21 illustrates a flowchart for network access cards for a disconnection in an example of the invention.

FIG. 22 illustrates a flowchart for network access cards for a connection in an example of the invention.

7

FIG. 23 illustrates a flow chart for network access cards for a disconnection in an example of the invention.

FIG. 24 illustrates a flowchart for global access in an example of the invention.

FIG. 25 illustrates a flowchart for an access server for user based proxies in an example of the invention.

FIG. 26 illustrates a flowchart for a user proxy for user based proxies in an example of the invention.

FIG. 27 illustrates a flowchart for an access server for subscriber based proxies in an example of the invention.

FIG. 28 illustrates a flowchart for a subscriber proxy for subscriber based proxies in an example of the invention.

FIG. 29 illustrates a flowchart for a dynamic proxy for dynamic proxies in an example of the invention.

FIG. 30 illustrates a block diagram of the access execution environment in an example of the invention.

FIG. 31 illustrates a flow chart for the access execution environment in an example of the invention.

FIG. 32 illustrates a flowchart for an access server for domain name scoping in an example of the invention.

FIG. 33 illustrates a flowchart for a database system for domain name scoping in an example of the invention.

FIG. 34 illustrates a flowchart for an access server for an inband domain name service lookup in an example of the invention.

FIG. 35 illustrates a flowchart for inline access service triggering in an example of the invention.

FIG. 36 illustrates a flowchart for an access server for access service triggering in an example of the invention.

FIG. 37 illustrates a flowchart for a database system for access service triggering in an example of the invention.

FIG. 38 illustrates a flow chart for the personal URL lookup in an example of the invention.

FIG. 39 illustrates a flow chart for the personal URL update in an example of the invention.

FIG. 40 illustrates a flowchart for an access server for auditing in an example of the invention.

FIG. 41 illustrates a flowchart for a database system for auditing in user predictive caching in an example of the invention.

FIG. 42 illustrates a flowchart for a database system for caching in user predictive caching in an example of the invention.

FIG. 43 illustrates a flowchart for a database system for auditing in user controlled caching in an example of the invention.

FIG. 44 illustrates a flowchart for a database system for caching in user controlled caching in an example of the invention.

FIG. 45 illustrates a flowchart for switching access by a user in an example of the invention.

FIG. 46 illustrates a flowchart for switching access by a service provider in an example of the invention.

FIG. 47 illustrates a flowchart for dynamic switching access in an example of the invention.

FIG. 48 illustrates a flowchart for an access server for network address failover in an example of the invention.

FIG. 49 illustrates a flowchart for a database system for network address failover in an example of the invention.

FIG. 50 illustrates a flowchart for an access server for network busy forwarding in an example of the invention.

FIG. 51 illustrates a flowchart for a database system for network busy forwarding in an example of the invention.

8

FIG. 52 illustrates a flowchart for an access server for a busy flag when the destination network device is busy in an example of the invention.

FIG. 53 illustrates a flowchart for an access server for forwarding, if the destination network device timeouts in an example of the invention.

FIG. 54 illustrates a flowchart for an access server for schedule alias resolution in an example of the invention.

FIG. 55 illustrates a flowchart for a database system for scheduled alias resolution in an example of the invention.

FIG. 56 illustrates a flowchart for an access server for destination controlled forwarding in an example of the invention.

FIG. 57 illustrates a flowchart for a database system for destination controlled forwarding in an example of the invention.

FIG. 58 illustrates a flowchart for an access server for network endpoint availability management in an example of the invention.

FIG. 59 illustrates a flowchart for a database system for network endpoint availability management in an example of the invention.

FIG. 60 illustrates a flowchart for a firewall/router for service capability monitor in an example of the invention.

FIG. 61 illustrates a service capability monitor software architecture for a service capability monitor in an example of the invention.

FIG. 62 illustrates a flowchart for the network logic for service capability monitor in an example of the invention.

FIG. 63 illustrates a flowchart for the service logic for service capability monitor in an example of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Communication Network Architecture—FIGS. 5 and 6

The following description and associated figures discuss specific examples intended to teach the present invention to those skilled in the art. Those skilled in the art will appreciate numerous variations from these examples that do not depart from the scope of the invention. Those skilled in the art will also appreciate that various features described below could be combined in various ways to form multiple variations of the invention.

FIG. 5 illustrates a network architecture 500 in an example of the invention. The network architecture 500 comprises a service network 530, a service network 540, and an access network 520. The access network 520 comprises a database system 522, an access server 524, a firewall/router 526, a firewall/router 556, and an access server 554. A user network 510 includes a network device 512. The user network 510 is connected to the access server 524. The access server 524 is connected to the firewall/router 526 and the database system 522. The firewall/router 526 is connected to the service network 530, the service network 540, and the database system 522. The firewall/router 556 is connected to the service network 530, the service network 540, the database system 522, and the access server 554. The access server 554 is connected to the database system 522 and the user network 510. The user network 560 comprises a network device 562 and a network device 564.

The access network 520 provides an interface between the user network 510 and 560 and the service networks 530 and

540. The interface function provides user access profiles, security, switching, and caching. The user network 510 and 560 could be a residential or business communication system that includes network devices. A network device 512, 562, and 564 could be any device configured to exchange data or information with the access network 520. Some examples of network devices are wireless and wireline telephones, computers, modems, servers, and/or data terminals, along with associated interconnections and network interfaces. For illustrative purposes in the examples below, a user interacts with the network device 512 to access services provided by the network architecture 500 and the user network 560. The user network 510 could also be a communication destination for other users. In these cases, the access network 520 provides destination performance monitoring and control. The example of a user interacting with the network device 562 and 564 to access services provided by the network architecture 500 and the user network 510 is not discussed below for the sake of clarity.

The access server 524 and the database system 522 could be adapted from components used in current ISPs, and the access network 520 could be integrated into these ISP components. The database system 522 houses user access profiles for the users in the user networks 510 and 560 with external access rights. The user access profiles provide a data clearinghouse for user-related information for security, service options, current state, and customized macros. The service networks 530 and 540 could be voice or data systems, such as the public telephone network, Internet, public data networks, and private data networks.

When a user requests access to services, the access network 520 processes the user access profile for the user. The access network 520 performs security measures to validate the user. The access network 520 then binds the user to a terminal and to a service. This user-terminal-service binding correlates the user's capabilities in the user access profile with the capabilities of the user's current terminal and the current capabilities of the service provider. The access network 520 activates a network shell for the user and the user's terminal. The network shell is an interface that is customized for the user and the terminal. The user invokes services using the network shell.

The user access profile may contain several macros that can be as simple as address translations or as complex as lengthy computer programs. The user's network shell provides access to the user's macros. The user access profile also stores caching instructions. The caching instructions control the collection and storage of information within the access network 520 for immediate access by the user.

FIG. 6 depicts an access network in an example of the invention. The access network 520 comprises the database system 522, the access server 524, the firewall/router 526, the firewall/router 556, and the access server 554. The database system 522 comprises a local database system 570, a Lightweight Directory Access Protocol (LDAP) interface system 571, a central database system 580, a local database system 590, and an LDAP interface system 591. The local database system comprises a user profile system 572, an audit database system 573, a cache database system 574, a host system 575, a security server 576, a user authorization system 577, an alias translation system 578, and a personal DNS system 579. The central database system 580 comprises a user authorization system 581, a financial interface 582, and a cross connect system 583. The local database system 590 comprises a user profile system 592, a user authorization system 593, and an availability system 594.

The access server 524 and the firewall/router 526 are connected to the local database system 570. The local

database system 570 is connected to the central database system 580 and the LDAP interface system 571. The central database system 580 is connected to the local database system 590. The local database system 590 is connected to the firewall/router 556, the access server 554, and the LDAP interface system 591.

An access provider is an entity that provides access to users who use communications services. A typical access provider comprises an access server, a firewall/router, and a local database system. The systems within the local database system 570 could be included within the local database system 590. Also, the systems within the local database system 590 could be included within the local database system 570. As discussed above, the user is accessing the network architecture through the user network 510. The duplication of systems in the local database system 570 and the local database system 590 is excluded for the purposes of clarity. A service provider is an entity that provides communication services to users who are accessing the service through an access provider.

Network User Access Profile—FIGS. 5, 6, 7, 8, 9, 10, 11, and 12

User Access Profile Inheritance

The user access profile is stored in the access network 520 and controls user access to services. The user access profile is any information or data associated with controlling user access to a service such as role identification, authorization, billing information and access preferences.

FIG. 7 depicts table for a user access profile in an example of the invention. A user access profile includes access information for the user, billing information, and preferences for access. Access information is any information or data related to providing the user access to the network architecture 500. Some examples of access information are user ID, password, name, account number, user alias, current network address, switching allowed flag, and other security information. Access information also include a list of services that the user has subscribed to or is allowed access to. In one embodiment, access information includes a cache of information that the user has accessed previously. Some examples of billing information are address and billing code including bank card numbers or prepaid account codes. Preferences for access allow the user to save choices or preferences to customize their access to the network architecture 400. Some examples of preferences for access are file formats and Quality of Service values. The user access profile may also include usage information such as time of day access, day of week, usages per day, usages per week, and usages per month.

Users typically set up their user access profiles when signing up for the access to the network architecture 500. In one embodiment, users create the user access profile through an inheritance process. Through the inheritance process, the user selects user profile information from other user access profiles in the network architecture 500. The user may inherit user profile information from user profile data structures such as templates, other user access profiles, the user's group or class, or other networks that the user is set up in. A user profile data structure is any user profile information to be retrieved when inheriting a user access profile.

In a prior solution, a command interpreter in a single computer operating system shell environment allows a user to customize the interpretation of command strings. The user may inherit portions of other user's shell environment by adding the other user's shell attributes. User access profiles are typically stored in the access provider's database. FIGS.

5 and FIG. 8 disclose one embodiment for inheriting user access profiles in an example of the invention. The user access profile is created through an inheritance process where the user is able to select capabilities, macros, functions, methods, and data to inherit from other profiles. In this embodiment, users inherit user access profiles from classes, groups, or provider recommendations. Features of user access profiles such as alias translation could then be implemented with new users rapidly. The inheritance of user access profiles also simplifies the configuration of users. In one example, a user initiates the user access profile inheritance by clicking a button on a website. Also, when a user requests a new service, the service provider automatically inherits the user access profile for the user so the user is able to use the requested service.

FIG. 8 depicts a flowchart for the access server 524 for inheriting a user access profile in an example of the invention. FIG. 8 begins in step 800. In step 802, the access server 524 waits for the next packet. The access server 524 then receives and processes the packet from the network device 562 in step 804. The access server 524 then checks if the destination is the user access profile in step 806. If the destination is not the user access profile, the access server 524 transmits the packet on the correct path in step 808 before returning to step 802. If the destination is the user access profile, the access server 524 then checks if the network device 562 is allowed access to the user access profile in step 1110. If the network device 512 is not allowed access to the user access profile, the access server 524 discards the packet and registers a network request security event in step 812 before returning to step 802.

If the network device 512 is allowed access to the user access profile, the access server 524 then checks if the user is allowed to update their user access profile in step 814. If the user is not allowed updates to their user access profile, the access server 524 generates and transmits a profile update not allowed message to the network device 512 in step 816 before returning to step 802. If the user is allowed updates to their user access profile, the access server 524 generates and transmits a user access profile update permission message asking if the user wishes to update the user access profile to the network device 512 in step 818. The access server 524 then checks if the user approved the user access profile update in step 820. If the user does not approve, the access server 524 generates and transmits a user access profile aborted message to the network device 512 in step 822 before returning to step 802.

If the user approves the user access profile update, the access server 524 sets an external request timer in step 824. The access server 524 then generates and transmits a user access profile update request to the database system 522 in step 826. A user access profile update request could be any signaling, message, or indication to update the user access profile. The access server 524 then checks if a reply was received or the external request timer expired in step 828. If the reply was not received and the external request timer did not expire, the access server 524 returns to step 828. If the reply was received or the external request timer did expire, the access server 524 checks if the reply was valid in step 830. If the reply was valid, the access server 524 generates an update complete message in step 832 before returning to step 802. If the reply was not valid, the access server 524 discards the packet and replies to the network device 512 that the user access profile update failed in step 834 before returning to step 802.

FIG. 9 depicts a flow chart for the database system 522 for inheriting a user access profile in an example of the inven-

tion. FIG. 9 begins in step 900. In step 902, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 904. In step 906, the database system 522 then checks if the access server requester is known. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 912. The database system 522 appends the reply to the packet and transmits the reply to the access server 524 in step 910. The database system 522 then replies with a decline message to the access server 524 in step 912 before returning to step 902.

If the requester is known, the database system 522 then checks whether the request is a user access profile update request in step 914. If the request is not a user access profile update request, the database system 522 checks if the request is a security event in step 916. If the request is not a security event, the database system 522 then registers an unknown action event in step 918 before returning to step 910. If the request is a security event, the database system 522 increments a path/device security record in step 924. The database system 522 then logs the path/device security record in step 926 before returning to step 902.

If the request is a user access profile update request, the database system 522 identifies and authenticates the user and the network device 512 in step 920. The database system 522 then checks if the access is valid in step 922. If the access is invalid, the database system 522 returns to step 924. If the access is valid, the database system 522 retrieves the user access profile information from a user profile data structure in step 928. The retrieval of user access profile information may be based on any information in the user access profile such as group or class or selections of inheritance user access profile presented to the user. The group or classes could be any logical grouping of user based on similar interests or situations such as work, family, and geographic location. The database system 522 then updates the user access profile in step 930 based on the user access profile information retrieved in step 928 and the user's selections for updating. The database system 522 then replies with an approve message to the access server 524 in step 932 before returning to step 902. In one embodiment of the invention, the database system 522 uses the user access profile system 572 to perform the operations disclosed in FIG. 9 and stores the user access profile in the user access profile system 572.

45 Network Shell

The user access profile includes data to provide a customized network shell to the user. The network shell is a user interface to the access network that is linked to programs, methods, macros, or service. Typically, configuration of the network shell is graphically presented to the user on a display. For example, the network shell appears as a list of alias selections that are associated with actions such as a program or macro for resources or services. An alias selection is any information that is associated with an action to be executed when the alias selection is selected. In another example, the alias selections are graphically presented as icons that relate to actions to be executed. The network shell overlays the standard DNS offered in IP networks, which reduces alias translation delays and required user keystrokes. In prior solutions, a command interpreter in an operating system shell environment allows a user to customize the interpretation of command strings, or a DNS translates "alias" values to addresses. FIGS. 5, 10, and 11 show one embodiment for network shells in an example of the invention.

FIG. 10 depicts a flowchart of an access server for executing a network shell in an example of the invention.

13

FIG. 10 begins in step 1000. In step 1002, the access server 524 waits for the next packet. The user selects an alias selection from the network shell graphically presented. In another embodiment, the user enters an alias value by a non-graphical means. Alternatively, the network shell is not presented to the user. The network device 512 transmits a packet including the alias selection to the access server 524. In step 1004, the access server 524 receives and processes the packet from the network device 512 and processes the packet. The access server 524 then checks if the network device 512 is allowed access to the network architecture 500. If the network device 512 is not allowed access to the network architecture 500, the access server 524 discards the packet and registers a network security event in step 1008 before returning to step 1002.

If the network device 512 is allowed access to the network architecture 500, the access server 524 checks if the user is recognized in step 1010. If the user is not recognized, the access server 524 returns to step 1008. If the user is recognized, the access server 524 retrieves the user's network shell in step 1012. In some embodiments, the user's network shell is retrieved from the user access profile in the access database 522 prior to presenting the network shell to the user. In step 1014, the access server 524 checks if the packet from the network device 512 includes an alias selection from the user's network shell. In one embodiment, specialized hardware is used to scan for aliases just as IP addresses are currently scanned for. If the packet does not include an alias selection from the user's network shell, the access server 524 proceeds to step 1018. If the packet does include the alias selection from the user's network shell, the access server 524 executes the action associated with the alias selection in step 1016 before returning to step 1002. In step 1018, the access server 524 processes the packet with the normal handling before returning to step 1002.

FIG. 11 depicts a flowchart of a database system for updating a network shell in an example of the invention. FIG. 11 begins in step 1120. In step 1122, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 1124. The database system 522 then checks if the access server requester is known in step 1126. If the access server requester is not known, the database system 522 registers an unknown requester event in step 1128. The database system 522 then appends the reply to the packet and transmits the packet to the access server 524 in step 1130. In step 1132, the database system 522 replies with a decline message to the access server 524 before returning to step 1122.

If the access server requester is known, the database system 522 checks if the request is a profile update for the network shell in step 1134. If the request is a profile update for the network shell, the database system 522 identifies and authenticates the user and the network device 512 in step 1136. The database system 522 then checks whether the access is valid in step 1138. If the access is invalid, the database system proceeds to step 1154. If the access is valid, the database system 522 retrieves the user access profile in step 1140. The database system 522 then updates the user access profile with the network shell with the user's alias selections and the associated programs, macros, functions or methods in step 1142. The database system 522 then replies with an approve message in step 1144 before returning to step 1122.

If the request is not a profile update, the database system 522 checks if the request is an action event in step 1146. If the request is an action event, the database system 522

14

performs the action and replies in step 1148 before returning to step 1122. If the request is not an action event, the database system 522 checks if the request is a security event in step 1150. If the request is not a security event, the database system 522 registers an unknown request type event in step 1152 before returning to step 1122. If the request is a security event, the database system 522 proceeds to step 1154. In step 1154, the database system increments a path/device security record. The database system 522 then appends the requester information to the packet and logs the event before returning to step 1122. In one embodiment of the invention, the database system 522 uses the user access profile system 572 to perform the operations disclosed in FIG. 11 and stores the user access profile in the user access profile system 572.

Service Based Directory

In prior systems, the RADIUS server provided the user with selections for transport modes. However, the user was not able to select available network based services. FIGS. 5 and 12 disclose one embodiment for a service based directory in an example of the invention. In this embodiment, access providers provide a list of services that the user can select. With the list of services, the access providers have the ability to advertise specific services to users. The list of services may be generated based on the user access profile to make the list user specific. Once the user makes the selection, the access server 524 connects the user to the service network such as Intranet, Internet, or private dedicated network that provides the selected service.

FIG. 12 depicts a flowchart for the services based directory in an example of the invention. FIG. 12 begins in step 1200. In step 1202, the access server 524 waits for the next connection from a network device in the user network 510. The access server 524 then receives a connection from the network device 512 in step 1204. Once the connection is established, the access server 524 generates and transmits a user ID query to the network device 512 in step 1206. The access server 524 then receives an ID reply and establishes a network device session in step 1208.

The access server 524 then generates an available services reply including a list of services in 1210. In one embodiment, the access server 524 generates the available services reply based upon information in the user access profile. The access server 524 receives a selected service reply from the network device 512 in step 1212. The access server 524 then connects the network device 512 to the selected service provider in step 1214. The access server 524 waits for the next packet in step 1214. The access server 524 then exchanges packets between the network device 524 and the selected service provider in step 1218.

Access Network User Binding—FIGS. 5, 13, 14, 15, and 16

User Access Profile Mobility

Users may access their user access profile from any network device connected to the network architecture 500. In a prior solution, a command interpreter in an operating system shell environment allows a user to customize the interpretation of command strings. In this environment, the access network identifies the user and the terminal device interface, which provides user mobility. The access network then executes customized actions for the user. FIGS. 5 and 13 show one embodiment of the invention for user access profile mobility. This embodiment provides user mobility in a distributed data network environment.

FIG. 13 depicts a flowchart of user access profile mobility in an example of the invention. FIG. 13 begins at step 1300. A user at the network device 512 signs on the access network

520 with their user ID. The network device 512 transmits user information with the user ID to the access server 524. In this embodiment, the user information is in the form of a packet. In step 1302, the access server 524 receives and processes the packet to check if the user access profile is local within the local database system 570. A user access profile is local within the local database system 570 when the user access profile is located in the local database system 570. If the user access profile is local, the access server 524 proceeds to step 1312.

If the user access profile is not local within the local database system 570, the access server 524 checks if the user ID is delimited with a provider ID in step 1304. One example of a user ID delimited with a provider ID includes a user's name and a provider ID separated by a delimiter such as joesmith@access.net. If the user ID is not delimited, the access server 524 retrieves the location of the default user access profile system using a default Lightweight Directory Access Protocol (LDAP) interface system 571 in step 1308 before proceeding to step 1312.

If the user ID is delimited, the access server 524 checks if the provider ID is valid in step 1306. If the provider ID is not valid, the access server 524 registers an invalid profile request security event before terminating at step 1326. If the provider ID is valid, the access server 524 uses the provider ID to retrieve the location of the local database system 590 from a foreign LDAP interface system 591 before proceeding to step 1312.

In step 1312, the access server 524 checks if the packet is a retrieve request for the user access profile. If the packet is a retrieve request, the access server 524 generates and transmits a request to retrieve the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then creates the user session ID and updates the user entries of the user access profile with the user ID and date/time updated in step 1312. The access server 524 then transmits a reply message with the session ID and the location of the LDAP server in step 1314 before terminating at step 1326.

If the packet is not a retrieve request, the access server 524 checks if the packet is a release request in step 1316. If the packet is a release request, the access server 524 retrieves the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then updates the user session ID and updates the user entries of the user access profile with the user ID and date/time updated in step 1318. The access server 524 then transmits a reply message with the session ID in step 1320 before terminating at step 1326.

If the packet is not a release request, the access server 524 checks if the packet is an update request in step 1322. If the packet is not an update request, the access server 524 registers an invalid profile request security event before terminating at step 1326. If the packet is an update request, the access server 524 retrieves the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then updates the user access profile with the information in the packet in step 1324. The access server 524 then transmits a complete message to the user network 510 to signify the profile update is complete before terminating at step 1326. In one embodiment, the local database system 570 uses the user access profile system 572 to retrieve and store the local user access profiles and the local database system 590 uses the user access

profile system 592 to retrieve and store the foreign user access profiles.

User, Device, and Service Sessions

Access network providers provide user and device sessions in addition to service sessions to distinguish users when providing communication services. A user session is the information associated with a user accessing a network. A device session is the information associated with a device being used to access a network. A service session is the information associated with a service being provided over a network. Service providers distinguish users instead of access devices or links. This allows multiple users to share a single access device with each user receiving their own customized or preferred services. Advantageously, service providers establish service access rights and restrictions such as preventing adult content for younger viewers or sharing an access device for business and personal use. Also, user, device, and service sessions allow a service provider to group multiple service providers to provide a composite of services to the user similar to a contractor/sub-contractor relationship.

FIGS. 5, 14, 15, and 16 disclose one embodiment for device, user, and service sessions in an example of the invention. FIG. 14 depicts a logical view of device, user, and service sessions in an example of the invention. Devices 1402, 1404, and 1406 are comprised of session type specific information and session links. Session type specific information include public keys, private keys, and session ID. Session links include user sessions, device sessions, and service sessions. The public keys and private keys are for encryption and decryption of messages. Session ID identifies the session ID for the device. User sessions are user sessions that the device is logically linked to. Service sessions are the service sessions the device is logically linked to. Users 1408, 1410, and 1412 are comprised of public keys, private keys, session ID, device sessions, and service sessions. Session ID identifies the session ID for the user. Device sessions are the device sessions the user is logically linked to. Service sessions are the service sessions the user is logically linked to. Services 1414, 1416, and 1418 are comprised of public keys, private keys, session ID, user sessions, device sessions, and sub service sessions. Session ID identifies the session ID for the service. User sessions are the user sessions the service is logically linked to. Device sessions are the device sessions the service is logically linked to. Sub service sessions are the service sessions the service is logically linked to.

In one example, device 1402 is linked to user B 1410 via a link 1422. User B is also linked to service N 1418 via a link 1424. Device 1402 contains the user information in the user session fields for user B 1410 and the service information in the service session fields for service N. User B 1410 contains the device information in the device session fields for device 1402 and the service information in the service session fields for service N 1418. Service N 1418 contains the device information in the device session fields for device 1402 and the user information in the user session fields for user B 1410. Service N 1418 is also linked to service A 1414 via a link 1426. Service N 1418 contains additional service information in the service session fields for service A 1414. Service A 1414 contains the composite service information in the service session fields for service N 1418. Service A 1414 also includes an owning service which is a reference to the service that owns service A.

FIG. 15 depicts a flowchart for a user based session in an example of the invention. FIG. 15 begins in step 1500. In step 1502, the network device 512 establishes a connection

17

with the access server 524. The access server 524 generates and transmits a user ID query to the user network 510 in step 1504. In step 1506, the network device 512 receives the user ID query and transmits a user ID reply to the access server 524. The access server 524 receives and processes the user ID reply to generate a device session. The network device 512 then transmits a packet to the access server 524. The access server 524 then receives the packet in step 1508. In step 1510, the access server 524 processes the packet to check if the packet includes a user session ID.

If the packet does not include the user session ID, the access server 524 transmits a login query encrypted by the network device's 512 public encryption key 1402 to the network device 512 in step 1512. The network device 512 decodes the login query with its public encryption key and transmits a login reply encrypted with its public encryption key in step 1516. The access server 524 then decodes the login reply with the private encryption key and checks if the user ID is valid. If the user is valid, the access server 524 generates a user session and a session ID in step 1520. In some embodiments, the user session ID is the original IP address. The access server 524 then encrypts with the public encryption key and transmits a complete reply with the user session ID to the network device 512 in step 1522 before returning to step 1508.

If the packet does include the user session ID, the access server 524 checks if the user and user session ID are valid in step 1514. If the user or user session ID is not valid, the access server 524 discards the packet and registers a security event in step 1515 before returning to step 1508. If the user and user session ID are valid, the access server 524 generates a service request with the user session ID and the service session ID if available. The access server 524 encrypts the service request with the public encryption key where appropriate. The access server 524 transmits the packet including the service request in step 1518 before returning to step 1508.

FIG. 16 depicts a flowchart for a service based session in an example of the invention. FIG. 16 begins in step 1600. The network device 512 transmits a service request to the access server 524. The access server 524 then receives the service request in step 1602. In step 1604, the access server 524 checks if the service request includes a service session ID.

If the service request does not include the service session ID, the access server 524 transmits a service ID query encrypted with the public encryption key to the network device 512 in step 1606. The network device 512 decodes the service ID query with its private key and transmits a service reply encrypted with the service public encryption key to the access server 524 in step 1610. The access server 524 then decodes the service reply with the service private encryption key and checks if the user is valid. If the user is valid, the access server 524 generates a service session and a session ID in step 1614. The access server 524 then transmits a complete reply with the service session ID to the network device 512 in step 1616 before returning to step 1602.

If the packet does include the service session ID, the access server 524 checks if the user and service session ID are valid in step 1608. In one embodiment, the service session ID is the destination IP address. If the user or service session ID is not valid, the access server 524 discards the packet and registers a security event before returning to step 1602. If the user and service session ID is valid, the access server 524 updates the service request with the user session ID and the service session ID. The access server 524

18

encrypts the service request with the service public encryption key where appropriate. The access server 524 transmits the service request with the user session ID and the service session ID in step 1612 before returning to step 1602.

Access Network Security—FIGS. 5, 17, and 18 Service Capability Firewall

A network service provider interfaces with a network access provider at a transport functional level. The interface typically includes Internet protocol firewalls to provide security. Unfortunately, the interface between the network service provider and the network access provider is not able to hide the implementation details such as addressing schemes, transport details, and equipment specifics. The hiding of implementation details is preferred to prevent hackers from manipulating the implementation details. FIGS. 5, 17 and 18 depict one embodiment for a service capability firewall in an example of the invention. In this embodiment, the interface between the network service provider and the network access provider occurs at a named functional level instead of the transport addressing level. A named function request is any request for a capability of service provided by the network service provider. This allows the network service provider to hide the implementation details. The network service provider exposes only functional capabilities to the users depending on their security rights.

FIG. 17 depicts a flowchart for the firewall 556 for service capability firewall in an example of the invention. FIG. 17 begins in step 1700. In step 1702, the firewall 556 waits for information. In this embodiment, the information is in the form of a packet. The firewall 556 receives and processes a packet including a named function request from the network device 512 in step 1704. The firewall 556 then checks whether the firewall 556 is the destination for the named function request in step 1706. If the firewall 556 is not the destination, the firewall 556 registers a network request security event in step 1710. In step 1712, the firewall 556 encapsulates the packet with path information and transmits the packet to the database system 522 before returning to step 1702.

If the firewall 556 is the destination, the firewall 556 checks whether the sending address is consistent with the path in step 1714. If the sending address is not consistent with the path, the firewall 556 returns to step 1710. If the sending address is consistent with the path and does not belong to the accessing network, the firewall 556 checks if the sending address/session ID/named function combination is cached in step 1716. If the sending address/session ID/named function combination is cached, the firewall 556 replaces the packet's named function request with the private cached destination address in step 1718. The firewall 556 then checks if the private destination address is known and allowed to pass in step 1720. If the destination is known and allowed to pass, the firewall 556 transmits the packet on the correct path with standard firewall access in step 1722 before returning to step 3802. If the destination is not known or not allowed to pass, the firewall 556 returns to step 1710.

If the sending address/session ID/named function combination is not cached, the firewall 556 set an external request timer in step 1724. The firewall 556 then generates and transmits a request for the database system 522 in step 1726. The firewall 556 checks whether a reply is received or the timer has expired in step 1728. If the reply is not received and the timer has not expired, the firewall 556 returns to step 1728. In another embodiment, a blocked I/O is used instead of the wait loop in step 1728. If the reply is received or the timer has expired, the firewall 556 checks if there is a valid

19

reply in step 1730. If the reply is invalid or no reply was received, the firewall 556 discards the packet and registers a translation failure event in step 1734 before returning to step 1702. If the reply is valid, the firewall 556 replaces the packet's named function request based on the reply and caches the address/session functions returned in step 1732 before returning to step 1720.

FIG. 18 depicts a flowchart for the database system 522 for service capability firewall in an example of the invention. FIG. 18 begins in step 1800. In step 1802, the database system 522 waits for the next request. The database system 522 then receives and processes the request in step 1804. In step 1806, the database system 522 then checks if the access server requester is known. If the access server requester is not known, the database system 522 registers an unknown requester event in step 1808. The database system 522 then appends the reply with the packet and replies to the access server 524 in step 1810. The database system 522 then generates and transmits a decline reply to the access server 524 in step 1812 before returning to step 1802.

If the requester is known, the database system 522 then checks if the request is a capability appeal in step 1814. A capability appeal is any message, signaling or instruction requesting a capability with a related network address in the service provider. If the request is not a capability appeal, the database system 522 then checks if the request is a security event in step 1816. If the request is not a security event, the database system 522 register an unknown request event in step 1817 before returning to step 1810. If the request is a security event, the database system 522 increments a path/device security record and appends the requester information to the request packet and logs the event in step 1822 before returning to step 1802. If the request is a capability appeal, the database system 522 then identifies the user and network device 512 in step 1818. The database system 522 then checks if the access is valid in step 1822. If the access is invalid, the database system 522 returns to step 1822. If the access is valid, the database system 522 retrieves the user and network device profiles in step 1824. The database system 522 then checks whether the capability is valid for the user and session state in step 1826. If the capability is invalid, the database system 522 returns to step 1822. If the capability is valid, the database system 522 generates a session ID, the network address of the capability requested, and functions available and appends the network address, which is only sent to the firewall/router 526 and not to the user, to the reply in step 1830. The database system 522 then transmits the approve reply to the access server 524 in step 1832 before returning to step 1802.

Access Network Service Authorization—

FIGS. 5, 19, 20, 21, 22, 23, and 24

Prepaid Access

Prepaid phone cards are commonly used in PSTN, where the customer pays a prepaid amount that is debited against when the customer makes a call. In data networks, prepaid cards are not currently being used. FIGS. 5 and 19 disclose one embodiment for prepaid access in an example of the invention. In this embodiment, users buy prepaid cards from network providers. When the user requests access to one of many access providers throughout the country, the access provider verifies the prepaid account code before providing the access. A prepaid account code is any number that relates to a user's prepaid account. The prepaid account is debited against for the charges related to the access. Other charges related to the service provided may also be debited against the prepaid account. For example, a user may purchase an item from a website and have the charges debited against the

20

prepaid account. Once the prepaid amount is reached, the access provider terminates the access to the user. In one embodiment, the network provider provides different levels of service such as gold, silver, and bronze. The gold service has guaranteed throughput but higher rates for access, while the bronze service has lower throughput and rates.

FIG. 19 depicts a flowchart for prepaid access in an example of the invention. FIG. 19 begins in step 1900. In step 1902, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 1904. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits information identifying a billing code. A billing code is any number that identifies a user for billing. Two examples of billing codes are prepaid account codes and credit card numbers. In this embodiment, the information identifying a billing code is a response including a prepaid account code to the login query. In other embodiments, the billing code is retrieved from the user access profile. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response including the prepaid account code to determine if the user is known in the local database system 520 in step 1906. A user is known when the user is allowed to use the access network 520. In one embodiment, the user is known in the local database system 520 if there is time/amounts left in the user's prepaid account. In another embodiment, the database system 522 evaluates a positive balance file to determine the remaining time/amount in the user's prepaid account. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 1908 before returning to step 1902.

If the user is not known, the database system 522 checks if there is an appeal server for user authentication in step 1910. In one embodiment, the access server 524 performs the appeal on a decline. If there is no appeal server, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2412 before returning to step 2402. If there is an appeal server, the database system 522 generates an authorization query including the prepaid account code for the central database system 580 in step 1914. The database system 522 checks if the user is known in the central database system 580 in step 1914. In one embodiment, the user is known in the central database system 580 if there is time/amounts left in the user's prepaid account. If the user is known, the database system 522 proceeds to step 1908. If the user is not known, the database system 522 proceeds to step 1910.

In one embodiment, the database system 522 uses a user authorization system 575 for checking if the user is known in the local database system 520. The user authorization system 575 contains all the prepaid customers, prepaid customer information, prepaid account codes, and the amount/quantity remaining in the prepaid account to verify if access is allowed. In one embodiment, the database system 522 uses a user authorization system 581 as the appeal server for checking if the user is known in the local database system 580. The user authorization system 581 contains all the prepaid customers, prepaid customer information, and the amount/quantity remaining in the prepaid account.

Bank Card Access

In prior systems, access providers authenticated users using their own databases. FIGS. 5, 20 and 21 disclose one

embodiment for bank card access in an example of the invention. In this embodiment, access providers authenticate users through bank card financial networks using the users' credit card numbers. Network providers use credit or debit card numbers as user ID and passwords for authentication and authorization purposes. Users use prepaid cards, phone cards, and credit cards in PSTN. However, no bank cards have been used for access to data networks other than for batch bill payment.

FIG. 20 depicts a flowchart for bank card access for a connection in an example of the invention. FIG. 20 begins in step 2000. In step 2002, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2004. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits information identifying a billing code. In this embodiment, the information identifying a billing code is a response including credit card numbers to the login query. In other embodiments, the billing code is retrieved from the user access profile. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response to see if the user is known in the local database system 570 in step 2006. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 2008 before returning to step 2002.

If the user is not known, the database system 522 processes the response to check if the user is identified by credit card numbers in step 2010. If the user is not identified by the credit card numbers, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2012 before returning to step 2002. If the user is identified by the credit card numbers, the database system 522 generates an authorization query as a pre-authorization hold or authorization/capture transaction for the central database system 580 in step 2014. In one embodiment, the central database system 580 uses a financial interface 582 to interface with a financial switch to banks for authentication and authorization. The database system 522 then checks if the user is authenticated and authorized by the central database system 580 in step 2016. If the user is authenticated and authorized, the database system 522 logs access information and an authorizing financial entity or institution in step 2018 before proceeding to step 2008. If the user is not known, the database system 522 checks if there is another database system for authorization such as for foreign user access in step 2018. If there is another database system, the database system 522 returns to step 2014. If there is not another database system, the database system 522 returns to step 2012.

FIG. 21 depicts a flowchart for bank card access for a disconnection in an example of the invention. FIG. 21 begins in step 2100. In step 2102, the access server 524 waits for a disconnection. The network device 512 then disconnects from the access server 524 in step 2104. The access server 524 then checks whether the user was on a pre-authorization hold in step 2106. If the user is not on a pre-authorization hold, the 522 logs access information and an authorizing database system 522 in step 2108 before proceeding to step 2102. If the user is on a pre-authorization hold, the database system 522 generates a pre-authorization complete transaction for the central database system 580 in step 2110 before returning to step 2108.

Access Cards

In prior systems, access providers authenticated users using their own databases. FIG. 5, 22 and 23 disclose one embodiment for network access cards in an example of the invention. An access card is a card that a user uses to access a network. The access card includes an access card account code. The access card account code is any number that relates to the user's access account. In this embodiment, access providers authenticate users who have access cards using other access providers' databases. Access providers then may enter into sharing agreements with other access providers. These sharing agreements allow users to use multiple access providers, which provide the users greater mobility and availability. Users use phone cards in PSTN for phone card access. However, no access cards for data networks have been used.

FIG. 22 depicts a flowchart for network access cards for a connection in an example of the invention. FIG. 22 begins in step 2200. In step 2202, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2204. In another embodiment of the invention, the user calls a toll free number to request access. A service control point is queried to determine where to route the request for access similar to an automatic call distribution (ACD). The request for access is then routed to the access server 524 to establish a connection between the network device 512 and the access server 524.

Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits a response including the access card account code to the login query. The access server 524 then transfers the response from the network device 512 to the database system 522. In this embodiment, the database system 522 checks if the user is allowed to use the access network 520 by checking whether the user is known. A user is known when the user is allowed to use the access network 520. For example, a user is known when their user access profile is located within the local database system 570 and their user access profile allows the user to use the access network 520. The database system 522 receives and processes the response including the access card account code to check if the user is known in the local database system 570 in step 2206. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 2208 before returning to step 2202.

If the user is not known, the database system 522 checks if the response included foreign network account information in step 2210. Foreign network account information is any information that is indicative of an account that is external to local database system 570 that the user is attempting to gain access. If there is no foreign network account information, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2212 before returning to step 2202.

If there is foreign network account information, the database system 522 identifies the local database system 590 based on the foreign network account information and generates an authorization query for the local database system 590 in step 2214. The database system 522 then checks if the user is authenticated and authorized by the local database system 590 in step 2216. If the user is authenticated and authorized, the database system 522 logs contract and settlements information returned by the local database system 590 or indicated by the database system 522 in relation to local database system 590 in step 2218 before proceeding to step 2208. If the user is not known, the

23

database system 522 proceeds to step 2212. In one embodiment, the local database system 570 uses the user authorization system 575 to check if the user is known in the local database system 570. In one embodiment, the local database system 590 uses the user authentication system 593 for authentication and authorization.

FIG. 23 depicts a flowchart for network access cards for a disconnection in an example of the invention. FIG. 23 begins in step 2300. In step 2302, the access server 524 waits for a disconnection. The network device 512 then disconnects from the access server 524 in step 2304. The access server 524 then generates and transmits a logoff query for the database system 522. The database system 522 then transfers the logoff query to the local database system 570 and the local database system 590. The database system 522 logs the access information and the authorizing database system in step 2308 before returning to step 2302.

Global Authentication

In prior systems, access providers authenticated users using their own databases. FIGS. 5 and 23 disclose one embodiment for global authentication in an example of the invention. In this embodiment, access providers authenticate users using a centralized database. Access providers then may enter into sharing agreements with other access providers. These sharing agreements allow users to use multiple access providers, which provide the users greater mobility. Currently, cellular telephone companies enter into these sharing arrangements for greater coverage. However, this sharing of databases for authentication and authorization has not occurred for data networks.

FIG. 24 depicts a flowchart for global access in an example of the invention. FIG. 24 begins in step 2400. In step 2402, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2404. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits a response to the login query. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response to see if the user is native in the local database system 570 in step 2406. In this embodiment, the database system 522 checks if the user is allowed to use the access network 520 by checking whether the user is native. A user is native when the user is allowed to use the access network 520. For example, a user is native when their user access profile is located within the local database system 570 and their user access profile allows the user to use the access network 520. If the user is native, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 4308 before returning to step 2402.

If the user is not native, the database system 522 checks if there is an authentication/authorization server in the database system 522 for a foreign network for user authentication in step 1910. If there is no authentication/authorization server in the database system 522 for the foreign network, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2412 before returning to step 2402. If there is an authentication/authorization server in the database system 522 for the foreign network, the database system 522 generates an authorization query for the central database system 580 in step 2414. The database system 522 then checks if the user is known in the central database system 580 in step 2414. If the user is known, the database

24

system 522 proceeds to step 2408. If the user is not known, the database system 522 proceeds to step 2410. In one embodiment, the central database system 580 uses a user authorization system 581 to check if the user is known.

Access Network Proxy/Environment—
FIGS. 5, 25, 26, 27, 28, 29, 30, and 31
User Based Proxy

A proxy is an application that represents itself as one or more network endpoints. The proxy receives a request for services from a user at a network endpoint and acts on behalf of the user in transmitting and receiving user requests and replies. There are Internet proxy agents that are user network access bind points. However, the Internet proxy agents are not user specific, and they act to protect user interests, not network. These proxy agents provide address translation and basic firewall functionality. Client focused proxies have extended to cookie collection and password handling.

FIGS. 5, 25 and 26 disclose one embodiment for user based proxies in an example of the invention. The user based proxies obtain information for the user and establishes a user specific network presence. A benefit of having a user specific network presence is that user access is handled by a process owned by a network security certificate authority that prevents Trojan horse network attacks. User based proxy provides a single control and monitor point for a user. The proxy agents provides a bind point for all user specific access such as user profile functionality, translations, security, and caching.

FIG. 25 depicts a flowchart for the access server 524 for user based proxies in an example of the invention. FIG. 25 begins in step 2500. In step 2502, the access server 524 waits for the next event. If the next event is a connection, the network device 512 connects with the access server 524 in step 2504. In step 2506, the access server 524 then collects user information and transfers the user information to the database system 522 to perform the security. In one embodiment, the database system 522 uses a security server 576 to perform the security. The access server 524 then checks if the user is allowed access in step 2508. If the user is not allowed access, the access server 524 disconnects the user in step 2510 before returning to step 2502.

If the user is allowed access, the access server 524 checks if a proxy is available in the database system 522 in step 2512. In one embodiment of the invention, the user proxies are in the host system 575. If the proxy is available, the access server 524 proceeds to step 2516. If the proxy is not available, the access server 524 starts the proxy in the database system 522 in step 2514 before proceeding to step 2516.

The access server 524 checks if the user access profile information is available in step 2516. If the user access profile information is not available, the access server 524 generates and transmits a user profile error to the network device 512 in step 2518 before returning to step 2502. If the user profile information is available, the access server 524 configures the proxy for the user in step 2520. The access server 524 then generates and transmits a message with the address of the user proxy and the public encryption key to the network device 512 in step 2522. The access server 524 also collects and transmits the access address of the network device 512 and the public encryption key to the user proxy. The access server 524 then returns to step 2502.

If the next event is a disconnection, the network device 512 disconnects with the access server in step 2524. The access server 524 then generates and transmits a reset command to the user proxy to clear the proxy state information and configuration in step 2526 before returning to

step 2502. If the next event is a status/request event from the user proxy, the access server 524 sets a user proxy reply timer in step 2528. The user proxy has written a status to the access server 524, and the user proxy receives a reply in step 2530 before returning to step 2502. If the next event is a timer expiration, the access server 524 receives the user proxy reply timer expiration in step 2532. The access server 524 then generates and transmits a continue wait message and status to the user proxy in step 2534 before returning to step 2528.

FIG. 26 depicts a flowchart for a user proxy for user based proxies in an example of the invention. FIG. 26 begins in step 2600. In step 2602, the user proxy waits for the next event. If the next event is the completion of the initialization, the user proxy checks if the initialization is complete in step 2604. The user proxy then sets a request timer in step 2606. The user proxy then request work from the access server 524 and transmits a status to the access server 524 in step 2608 before returning to step 2602. If the next event is the expiration of the request timer, the user proxy checks if the request timer expired in step 2610. The user proxy then registers a timeout in step 2612 before returning to step 2606.

If the next event is an access server 524 reply, the user proxy receives the access server 524 reply in step 2614. The user proxy then checks if the reply is a continue in step 2616. If the reply is a continue, the user proxy returns to step 2606. If the reply is not a continue, the user proxy processes the configuration information or action in step 2618 before returning to step 2606. If the next event is user request, the user proxy receives the user request for service in step 2620. The user proxy then checks if the user is valid in step 2622. If the user is valid, the user proxy then checks if the request is valid in step 2624. If the request is valid, the user proxy packages the request with the security certification and encryption in step 2626. The user proxy then transmits the request to the appropriate network destination in step 2628 before returning to step 2602. If the user or request is not valid, the user proxy registers a security event in step 2630 before returning to step 2602. The user proxy exchanges user information between the user network 510 and the appropriate network destination.

Subscriber Based Proxy

Another type of proxy is a subscriber based proxy. A subscriber is a logical entity such as an organization, corporation, or other grouping of users that has subscribed to services with a service provider. FIGS. 5, 27 and 28 disclose one embodiment for subscriber based proxies in an example of the invention. The subscriber based proxies obtain information for a user of a subscriber group and establishes a subscriber specific network presence. The benefit of having a subscriber specific network presence is that user access rights of the subscriber can be handled as a group, and group rights are owned by a network security certificate authority. The subscriber proxy agents provide a bind point for all subscriber specific access such as user profile functionality, translations, security, and caching.

FIG. 27 depicts a flowchart for the access server 524 for subscriber based proxies in an example of the invention. FIG. 27 begins in step 2700. In step 2702, the access server 524 waits for the next event. If the next event is a connection, the network device 512 connects with the access server 524 in step 2704. In step 2706, the access server 524 then collects user information and transfers the user information to the database system 522 to perform the security. In one embodiment, the database system 522 uses a security server 574 to perform the security. The access server 524

then checks if the user is allowed access in step 2708. If the user is not allowed access, the access server 524 disconnects the user in step 2710 before returning to step 2702. If the user is allowed access, the access server 524 checks if all required subscriber proxies are available in the database system 522 in step 2712. In one embodiment of the invention, the subscriber proxies are in the host system 575. If the proxies are available, the access server 524 proceeds to step 2716. If the proxies are not available, the access server 524 starts the required proxies in the database system 522 in step 2714 before proceeding to step 2716.

In step 2716, the access server 524 checks if the subscriber access profile information is available. If the subscriber access profile information is not available, the access server 524 generates and transmits a subscriber profile error to the network device 512 in step 2718 before returning to step 2702. If the subscriber profile information is available, the access server 524 configures the subscriber proxies for the subscriber in step 2720. The access server 524 then generates and transmits a message with the address of the subscriber proxy and the public encryption key to the network device 512 in step 2722. The access server 524 also collects and transmits the access address of the network device 512 and the public encryption key to the subscriber proxies. The access server 524 then returns to step 2702.

If the next event is a disconnection, the network device 512 disconnects with the access server in step 2724. The access server 524 then generates and transmits a reset command to the subscriber proxy in step 2726 before returning to step 2702. If the next event is a status/request event from the subscriber proxy, the access server 524 sets a subscriber proxy reply timer in step 2728. In step 2730, the subscriber proxy has written a status to the access server 524, and the subscriber proxy receives a reply before returning to step 2702. If the next event is a subscriber proxy timer expiration, the access server 524 receives the subscriber proxy reply timer expiration in step 2732. The access server 524 then generates and transmits a continue wait message and a status to the subscriber proxy in step 2734 before returning to step 2730.

FIG. 28 depicts a flowchart for a subscriber proxy for subscriber based proxies in an example of the invention. FIG. 28 begins in step 2800. In step 2802, the subscriber proxy waits for the next event. If the next event is the completion of the initialization, the subscriber proxy checks if the initialization is complete in step 2804. The subscriber proxy then sets a request timer in step 2806. The subscriber proxy then request work from the access server 524 and transmits a status to the access server 524 in step 2808 before returning to step 2802. If the next event is the expiration of the request timer, the subscriber proxy checks if the request timer expired in step 2810. The subscriber proxy then registers a timeout in step 2812 before returning to step 2806.

If the next event is an access server 524 reply, the subscriber proxy receives the access server 524 reply in step 2814. The subscriber proxy then checks if the reply is a continue in step 2816. If the reply is a continue, the subscriber proxy returns to step 2806. If the reply is not a continue, the subscriber proxy processes the configuration information or action in step 2818 before returning to step 2806. If the next event is a user request for service, the subscriber proxy receives the user request in step 2820. The subscriber proxy then checks if the user is valid in step 2822. If the user is valid, the subscriber proxy then checks if the request is valid in step 2824. If the request is valid, the subscriber proxy packages the request with the security

certification and encryption in step 2826. The subscriber proxy then transmits the request to the appropriate network destination in step 2828 before returning to step 2802. If the user or request is not valid, the subscriber proxy registers a security event in step 2830 before returning to step 2802. The subscriber proxy exchanges user information between the user network 510 and the appropriate network destination.

Dynamic Proxies

One prior system named "pluxy" allows a manual extension of a proxy. Pluxy requires manual determination and loading of a dynamic set of services, which is done on an operator basis. FIGS. 5 and 29 disclose one embodiment for dynamic proxies in an example of the invention. In this embodiment, both the user based proxies and the subscriber based proxies are extended in response to user actions. The dynamic proxy can be modified and enhanced to evolve with new services and/or protocols. The dynamic proxy size is smaller and more efficient by supporting only the logic that is being used. Implementation and development times of new service and protocol are also reduced because new proxies are not required with the new service and/or protocol.

FIG. 29 depicts a flowchart for a dynamic proxy for dynamic proxies in an example of the invention. FIG. 29 begins in step 2900. In step 2902, the dynamic proxy waits for the next event. If the next event is the completion of the initialization, the dynamic proxy checks if the initialization is complete in step 2904. The dynamic proxy then sets a request timer in step 2906. The dynamic proxy then request work from the access server 524 in step 2908 before returning to step 2902. If the next event is the expiration of the request timer, the dynamic proxy checks if the request timer expired in step 2910. The dynamic proxy then registers a timeout in step 2912 before returning to step 2906.

If the next event is an access server 524 reply, the dynamic proxy receives the access server 524 reply in step 2914. The dynamic proxy then checks if the reply is a continue in step 2916. If the reply is a continue, the dynamic proxy returns to step 2906. If the reply is not a continue, the dynamic proxy processes the configuration information in step 2918 before returning to step 2902. If the next event is a user request, the dynamic proxy receives the user request in step 2920. The dynamic proxy then checks if the user and the request are valid in step 2922. If the user and the request are valid, the dynamic proxy checks if there is a new service or protocol requested in step 2924. If there is a new service or protocol, the dynamic proxy proceeds to step 2934. If there is not a new service or protocol, the dynamic proxy packages the request with the security certification and encryption in step 2926. The dynamic proxy then transmits the request to the appropriate network destination in step 2928 before returning to step 2902. If the user or request is not valid, the subscriber proxy registers a security event in step 2930 before returning to step 2902.

If the next event is a new service/protocol request, the dynamic proxy receives the new service/protocol request in step 2932. In step 2934, the dynamic proxy processes the new service/protocol request to generate a handler request for a new service/protocol handler in step 2934 before returning to step 2926. The dynamic proxy receives the handler and executes the handler for the new service or proxy. In one embodiment, the dynamic proxy is enhanced by loading apple-like extensions from a handler system similar to a web browser.

Access Execution Environment

Proxy servers in the Internet represents a user in one network as existing in another network to obtain services for

the user. One problem with Internet proxy servers is the absence of the management of network resource utilization. In TINA-C, the User Agent and the User Session Manager are restricted to known service binding because a new service requires knowledge of or logic for a new CORBA interface. One problem with TINA-C is the absence of any provider resource management capabilities. Unfortunately, both Internet proxy servers and TINA-C do not provide any means of associating or viewing all network resources used by a user access session. Another problem is the lack of securing provider network resource against tampering by accessing users. Also, Internet proxy servers and TINA-C do not provide a platform-supported means of enforcing security levels on network access users. Both system also lack a validation of potential execution capability extensions.

FIGS. 5, 30, and 31 disclose one embodiment for an access execution environment in an example of the invention. The access execution environment easily and efficiently manages and secures network access by providing an execution environment in the access provider environment. Users access shared services and resources through their specific access execution environment. Because the access execution environment is in the access provider environment, the access provider can view the all of the user's activity by observing the execution environment. In this embodiment, the execution environment is setup as a standard system user on the network access platform, which allows management of the system user by operating system level user resource controls, quotas, and management mechanisms.

The access execution environment includes a multi-tasking virtual machine, a script interpreter, alias resolution capabilities, security certificate authentication, configuration handler, session caching, and protocol handling components. FIG. 30 depicts a block diagram of the access execution environment in an example of the invention. In FIG. 30, a network access platform 3000 comprises an execution environment manager 3010, a program and attribute database 3020, an execution environment A 3030, a transport A 3032, an execution environment B 3040, a transport B 3042, an execution environment C 3050, a transport C 3052, an execution environment D 3060, a transport D 3062, an execution environment E 3070, and a transport E 3072. In one embodiment, the network access platform 3000 is included within the database system 522. In another embodiment, the network access platform is included within the host system 575. The transport A 3032 is connected to the execution environment A 3030. The transport B 3042 is connected to the execution environment A 3040 and a workstation 3080. The transport C 3052 is connected to the execution environment C 3050. The transport D 3062 is connected to the execution environment D 3060. The transport E 3072 is connected to the execution environment E 3070. The execution environment A 3030 includes a transport handler 3034, a configuration handler 3036, and a security handler 3038. The execution environment B 3040 includes a transport handler 3044, a configuration handler 3046, and a security handler 3048. The execution environment C 3050 includes a transport handler 3054, a configuration handler 3056, and a security handler 3058. The execution environment D 3060 includes a transport handler 3064, a configuration handler 3066, and a security handler 3068. The execution environment E 3070 includes a transport handler 3074, a configuration handler 3076, and a security handler 3078.

In operation, a system administrator sets up a number of network access user accounts on the hardware platform. The administrator then sets up resource limits for each account/

platform resource. The administrator then starts an execution environment **3030**, **3040**, **3050**, **3060**, and **3070** for each access user account ID. The execution environment **A 3030** initializes and loads a configuration handler **3036**. The execution environment **A 3030** then loads the security handler **3038** and the transport handler **3034**. The transport handler **3034** opens a logical or physical transport port **A 3032** on the hardware platform using port information from the configuration handler.

FIG. 31 depicts a flow chart for the access execution environment in an example of the invention. FIG. 31 begins in step **3100**. In step **3102**, the security handler **3038** waits for a connect message from the transport handler **3034**. The security handler **3038** then receives the connect message from the transport handler **3034** in step **3104**. The security handler **3038** then generates and transmits a logon request via the transport handler **3034** in step **3106**. The security handler **3038** then checks if a reply for the logon request from the transport handler **3034** has been received in step **3108**. If the reply for the logon request has not been received, the security handler **3038** checks if a reply timeout has occurred in step **3110**. If a reply timeout has not occurred, the security handler **3038** returns to step **3108**. If a reply timeout has occurred, the security handler **3038** generates and transmits a disconnect message to the transport handler **3034** in step **3112** and returns to step **3102** to wait for a connect message.

In step **3114**, the security handler **3038** receives the reply for the logon request. Then the security handler **3038** retrieves the location of a security server from the configuration handler **3036** and transmits the logon information to the security server in step **3116**. The security handler **3038** then checks if a reply to the logon information from the security server has been received in step **3118**. If no reply has been received, the security handler **3038** checks if a reply timeout has occurred in step **3120**. If a reply timeout has not occurred, the security handler **3038** returns to step **3118**. If a reply timeout has occurred, the security handler **3038** checks if the logon information has been sent three times in step **3122**. The number of tries could be configurable. If the logon information has not been sent three times, the security handler **3038** returns to step **3116** to transmit the logon information. If the logon information has been sent three times, the security handler **3038** returns to step **3112** to send a disconnect message.

If the security server replied before the reply timeout, the security handler **3038** checks if the reply is an accept message in step **3124**. If the reply is a decline message, the security handler **3038** returns to step **3112**. If the reply is an accept message, the security handler **3038** transmits the accept message configuration parameters to the configuration handler **3036** in step **3126**. The configuration handler **3036** then loads attributes and programs and executes programs specified by the security server reply in step **3128**. The execution environment **A 3030** performs the execution of programs for the user in step **3130**. In another embodiment, the programs request attributes and/or programs to be loaded and/or executed. The programs include the ability to interface with users via the transport handler and load/execute other programs required by request types.

The configuration handler **3036** then checks if the transport handler **3034** receives a disconnect message in step **3132**. If the transport handler **3034** has not received a disconnect message, the configuration handler **3036** returns to step **3130**. If the transport handler **3034** has received a disconnect message, the transport handler **3034** transmits the disconnect message to the configuration handler **3036** in step

3134. In step **3136**, the configuration handler **6536** based on port configuration attributes closes the transport port **3032**, resets the transport handler **3034**, and notifies the execution environment manager **3010** to create a new execution environment for that port. The configuration handler **3036** gracefully shuts down all handlers, programs, and eventually the execution environment **A 3030**. The configuration handler **3036** transmits a shutdown message to the execution environment manager **3010** in step **3138** before terminating in step **3140**. The execution environment manager **3010** restarts the execution environment **A 3030** upon notification of the shutdown. The operations of the other execution environments **3040**, **3050**, **3060**, and **3070** perform in the same manner as the execution environment **A 3030** and are not discussed for the sake of clarity.

Access Network Translations—FIGS. 5, 32, 33, 34, 35, 36, 37, 38, and 39

Domain Name Scoping

The typing in of domain names such as www.nypostonline.com is quite cumbersome for the user to access specific services. The Domain Name Server (DNS) translates the domain name to a network address for the service. Clicking on bookmarks or favorites in browsers, which reference the domain names, does eliminate the need to type in the domain names. However, when users change network devices, these bookmarks or favorites do not follow the user. In an operating system environment, the command interpreter shell allows the user to customize interpretation of command strings. If no customization is loaded, the command interpreters interprets in a default fashion.

FIGS. 5, 32, and 33 disclose one embodiment for domain name scoping in an example of the invention. This embodiment provides a customizable network shell to overlay the standard DNS service offered in Internet Protocol based networks. Users then are able to define user specific acronyms or aliases for specific or logical services. An acronym or alias indicates domain names, macros, programs, actions, or network addresses. If the translation for the alias does not exist in the list of aliases for the user, the access server **524** checks if the alias exists in the list for a group in which the user belongs. The access server **524** then checks if the alias exist in the DNS for the general network. In another embodiment, the database system **522** checks the existence of the alias in the list for the group and in the DNS for the general network. There are numerous variations in the hierarchy for searching for an alias but are not discussed for the sake of simplicity.

FIG. 32 depicts a flowchart for the access server **524** for domain name scoping in an example of the invention. FIG. 32 begins in step **3200**. In step **3202**, the access server **524** waits for a packet. In this embodiment, the access server **524** receives information in the form of packets from the network device **512**. The access server **524** then receives and processes the packet from the network device **512** in step **3204**. The access server **524** then checks if the packet is an alias translation request in step **3206**. If the packet is not an alias translation request, the access server **524** transmits the packet with standard access logic in step **3214** before returning to step **3202**.

If the packet is an alias translation request, the access server **524** then checks if the alias translation is cached for the user in step **3208**. If the alias translation is cached, the access server **524** reformats the packet using the cached alias translation in step **3216** before returning to step **3214**. If the alias translation is not cached, the access server **524** sets an external request timer in step **3210**. The access server **524** then generates and transmits an alias translation request to

the database system 522 in step 3212. The alias translation request is any message, instruction, or signaling indicative of requesting an alias translation. The access server 524 checks whether a reply has been received or the external request timer has expired in step 3218. If a reply has not been received and the external request timer has not expired, the access server 524 returns to step 3218.

If a reply has been received or the external request timer has expired, the access server 524 checks if the reply was valid in step 3220. If the reply is invalid, the access server 524 discards the packet and replies with an invalid alias message or alias error message to the network device 512 before returning to step 3202. If the reply is valid, the access server 524 caches the alias translation for the user in step 3222 before returning to step 3216.

FIG. 33 depicts a flowchart for the database system 522 for domain name scoping in an example of the invention. FIG. 33 begins in step 3300. In step 3302, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 3304. In step 3306, the database system 522 then checks if the access server requester is known in step 3306. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 3308. The database system 522 appends the reply information to the packet in step 3310. The database system 522 replies with a decline message to the access server 524 in step 3312 before returning to step 3302.

If the access server requester is known, the database system 522 then checks whether the request is an alias translation request in step 3314. If the request is not an alias translation request, the database system 522 registers an unknown request event in step 3316 before returning to step 3310. If the request is an alias translation request, the database system 522 identifies and authenticates the user and the network device 512 in step 3318. If the access is invalid, the database system 522 increments a path/device security record in step 3322. The database system 522 then appends the requester denial information to the request packet in step 3324. The database system 522 then replies with a decline message to the access server 524 in step 3312 before returning to step 3302.

If the access is valid, the database system 522 retrieves the user's alias translation list in step 3326. The database system 522 then translates the alias for the user and appends the translation to a reply to the access server 524. The database system 522 then replies with an approve message to the access server 524 in step 3330. In one embodiment of the invention, the database system 522 uses the alias translation system 578 to perform the operations disclosed in FIG. 33 and stores the aliases and alias translations in the alias translation system 578.

Inband Domain Name Service Lookup

The Domain Name Server (DNS) translations of a domain name to a network address delays user interaction with the service provider. In prior solutions in voice telephone Signaling System #7 networks, a signal transfer point performs an inband local number portability (LNP) lookup on a ISDN part call setup request such as an Initial Address Message (IAM). This inband lookup eliminates the network switch from launching a Transaction Capabilities Application Part (TCAP) LNP query to translate the telephone number in the IAM.

FIGS. 5 and 34 disclose one embodiment for inband domain name service in an example of the invention. This embodiment provides a cache for the access server 524 for alias translations on a per user basis. Thus, the external

queries for translations of aliases to domain names, macros, programs, or network addresses are eliminated. If the translated value of a request is in the alias translation cache, no translation request will be required to a DNS server. Therefore, users experience reduced delays when requesting a service.

FIG. 34 depicts a flowchart for the access server 524 for an inband domain name service lookup in an example of the invention. FIG. 34 begins in step 3400. In step 3402, the access server 524 waits for a packet. The access server 524 then receives and processes the packet from the network device 512 in step 3404. The access server 524 then checks if the destination network address and user is valid in step 3406. If the destination network address and user is valid, the access server 524 transmits the packet on the correct path to reach the destination in step 3408 before returning to step 3402.

If the destination network address or the user is invalid, the access server 524 then checks if the alias translation is cached for the user in step 3410. If the alias translation is cached, the access server 524 reformats the packet using the cached alias translation in step 3412 before returning to step 3402. If the alias translation is not cached, the access server 524 sets an external request timer in step 3414. The access server 524 then generates and transmits an alias translation request to the database system 522 in step 3416. The access server 524 checks whether a reply has been received or the external request timer has expired in step 3418. If a reply has been received and the external request timer has not expired, the access server 524 returns to step 3418.

If a reply has been received or the external request timer has expired, the access server 524 checks if the reply was valid in step 3420. If the reply is invalid, the access server 524 discards the packet and replies with an invalid alias message to the network device 512 in step 3424 before returning to step 3402. If the reply is valid, the access server 524 caches the alias translation for the user in step 3422 before returning to step 3412.

Inline Access Service Triggering

Typical firewalls filter packets out on a request by request basis on "what is not allowed". FIGS. 5 and 35 disclose one embodiment for inline access service triggering. In this embodiment, the access server 554 filters packets out on a "what is allowed" basis. Also, the access server 554 performs packet filtering on a service access basis.

FIG. 35 depicts a flowchart for inline access service triggering in an example of the invention. FIG. 35 begins in step 3500. In step 3502, the access server 554 waits for the next packet. In this embodiment, the access server 554 receives information in the form of packets from the network device 512. The access server 554 receives and processes a packet from the network device 512 in step 3504. In step 3506, the access server 554 checks if the protocol, sending address, and the destination address are known and allowed to pass. In one embodiment, the access server 554 checks the database system 522 for the determination made in step 3506. If the protocol, sending address, and the destination address are known and allowed to pass, the access server 554 checks if the sending address is consistent with the path in step 3508. If the sending address is not consistent with the path, the access server 554 registers a path security event in step 3510. The access server 554 then formats the packet with path information in step 3512. The access server 554 also increments the path/device security record and logs the event before returning to step 3502. If the sending address is consistent with the path, the access server 554 transmits the packet on the correct path with standard firewall access in step 3514 before returning to step 3502.

If the protocol, sending address, or the destination address are not known or not allowed to pass, the access server 554 checks if the request is a filter appeal in step 3515. If the request is not a filter appeal, the access server 554 discards the packet in step 3526 before returning to step 3502. If the request is a filter appeal, the access server 554 identifies and authenticates the user and the network device 512 in step 3516. The access server 554 then checks if the access is valid in step 3518. If the access is invalid, the access server 554 returns to step 3510. If the access is valid, the access server 554 retrieves the user access profile and network device's 512 profile in step 3520. The access server 554 then modifies the access logic for filter modification in step 3524. The access server 554 then modifies the protocol and address filters based on the request in step 4824 before returning to step 3514. In some embodiments, the access server 554 modifies the filters in conformance with the user access profile.

Access Service Triggering

Access providers sometimes need to extend their authentication logic beyond their primary access devices. No prior system in data networks extends the authentication logic beyond what is performed by the access provider's access devices. FIGS. 5, 36 and 37 disclose one embodiment for access service triggering. In this embodiment, the access server 524 triggers a request to external access control logic. Access providers extend the access and authentication logic beyond their access devices while maintaining centralized control of the authentication logic and user access profiles. Also, the access server 554 performs packet filtering on a service access basis.

FIG. 36 depicts a flowchart for the access server 524 for access service triggering in an example of the invention. FIG. 36 begins in step 3600. In step 3602, the access server 524 waits for the next packet. The access server 524 then receives and processes a packet from the network device 512 in step 3604. In step 3606, the access server 524 checks if the protocol, sending address, and the destination address are known and allowed to pass. In one embodiment, the access server 524 checks the database system 522 for the determination made in step 3606. If the protocol, sending address, and the destination address are known and allowed to pass, the access server 524 checks if the sending address is consistent with the path in step 3608. If the sending address is not consistent with the path, the access server 524 registers a path security event in step 3610. The access server 524 then formats the packet with path information in step 3612 before returning to step 3602. If the sending address is consistent with the path, the access server 524 transmits the packet on the correct path with standard firewall access in step 3614 before returning to step 3602.

If the protocol, sending address, or the destination address are not known or are not allowed to pass, the access server 524 sets an external request timer in step 3616. The access server 524 then generates and transmits a request with the path information to the database system 522 in step 3618. The access server 524 then checks if a reply has been received or the external request timer has expired in step 3620. If the reply has not been received or the external request timer has not expired, the access server 524 checks if the reply is valid in step 3622. If the reply is invalid, the access server 524 discards the packet in step 3624 before returning to step 3602. If the reply is valid, the access server 524 then modifies the protocol and address filters based on the reply in step 3624 before returning to step 3614.

FIG. 37 depicts a flowchart for the database system 522 for access service triggering in an example of the invention.

FIG. 37 begins in step 3700. In step 3702, the database system 522 waits for the next request. The database system 522 then receives and processes the request in step 3704. In step 3706, the database system 522 then checks if the access server requester is known in step 3706. If the access server requester is not known, the database system 522 registers an unknown requester event in step 3708. The database system 522 then appends the reply to the packet in step 3710. The database system 522 then generates and transmits a decline reply to the access server 524 in step 3712 before returning to step 3702.

If the access server requester is known, the database system 522 then checks if the request is a filter appeal in step 3714. If the request is not a filter appeal, the database system 522 then checks if the request is a security event in step 3716. If the request is not a security event, the database system 522 registers an unknown action event in step 3717 before returning to step 3712. If the request is a security event, the database system 522 increments a path/device security record, appends the requester information to the request packet, and logs the event in step 3718. The database system 522 then checks if the request is a security event in step 3719. If the request is not a security event, the database system 522 then returns to step 3712. If the request is a security event, the database system 522 proceeds to step 3728.

If the request is a filter appeal, the database system 522 then identifies the user and network device 512 in step 3720. The database system 522 then checks if the access is valid in step 3722. If the access is invalid, the database system 522 returns to step 3718. If the access is valid, the database system 522 retrieves the user and network device profiles in step 3724. The database system 522 then generates access logic and appends the access logic to a reply in step 3726. The database system 522 then transmits the approve reply to the access server 524 in step 3728 before returning to step 3702.

Personal URL

The Internet currently uses Uniform Resource Locator (URL) addresses such as www.yahoo.com so that the address is in more human readable form than the network address. Unfortunately, user cannot distinguish themselves by a network address because the user's network address changes when the user's access point changes. FIGS. 5, 38 and 39 disclose one embodiment for a personal URL. In this embodiment, a network user may publish their location on the network by a user alias. A user alias is any alias that relates to the network address of a user. Logically linking a user's current network address with a user alias allows a user to be located wherever the user accesses the network.

FIG. 38 depicts a flowchart for a personal URL lookup in an example of the invention. FIG. 38 begins in step 3800. In step 3802, the database system 522 waits for the next packet. The database system 522 then receives and processes the packet including the user alias. The database system 522 then checks if a user alias translation is cached for this user in step 3806. If the user alias translation is cached, the database system 522 replies with the current network address of the user in step 3808 before returning to step 3802.

If the user alias translation is not cached, the database system 522 sets an external request timer in step 3810. The database system 522 then generates and transmits a request for user alias translation logic in step 3812. In one embodiment, the database system 522 transmits the request for user alias translation logic to the database system that contains the user access profile. The database system 522

then checks whether a reply was received or the external request timer expired in step 3814. If the reply was not received and the external request timer did not expire, the database system 522 returns to step 3814. If a reply was received or the external request timer did expire, the database system 522 checks if the reply was valid in step 3816. If the reply was invalid, the database system 522 replies with an invalid name message in step 3824 before returning to step 3802.

If the reply was valid, the database system 522 checks if there is a current network address for the user in step 3818. If there is no current network address for the user, the database system 522 replies with user alias not currently in network message in step 3820 before returning to step 3802. If there is a current network address for the user, the database system 522 caches the user alias translation in step 3822 before returning to step 3808.

FIG. 39 depicts a flowchart for a personal URL update in an example of the invention. FIG. 39 begins in step 3900. In step 3902, the database system 522 wait for a request. The database system 522 then receives and processes the request to update the user alias in step 3904. In one embodiment, the database system receives the request from a database system that contains the user access profile. In another embodiment, the request comes from the access server 524. The database system 522 then checks if the requester is known in step 3906. If the requester is not known, the database system 522 registers an unknown requester event in step 3910. The database system 522 then transmits a reply with decline in step 3912 before returning to step 3902.

If the requester is known, the database system 522 then checks if the requester is allowed to update the user alias in step 3914. If the requester is not allowed, the database system 522 registers an unauthorized requester event in step 3916 before returning to step 3912. If the requester is allowed to update, the database system 522 then identifies and authenticates the user and current network address in step 3918. The database system 522 then checks if the access is valid in step 3920. If the access is invalid, the database system 522 increments the path/device security record in step 3922. The database system 522 also appends the requester information to the request and logs the event before returning to step 3912.

If the access is valid, the database system 522 then updates the user alias translation with the current network address in step 3924. The database system 522 then replies with an approve in step 3926 before returning to step 3902. In one embodiment of the invention, the database system 522 uses the personal DNS system 579 to perform the operations disclosed in FIGS. 38 and 39.

Access Network Caching—FIGS. 5, 40, 41, 42, 43, and 44

Predictive Caching

Users that are dialed into the network at a rate of 56 kbps or greater typically retrieve information at a rate of 2–4 kbps. Various equipment between the network access provider and the service provider cause this lower rate of transmission. Some examples of the equipment are network access provider equipment, service provider equipment, network backbone overloading, traffic shaping device, firewalls, and routers. One solution for compensating for the lower rate of transmission is caching. Computers typically contain a memory cache for specific application or devices. Firewalls and routers may also contain caches for data. Unfortunately, none of these caches are user specific.

FIGS. 5, 40, 41 and 42 disclose one embodiment of the invention for predictive end user caching. Predictive end

user caching advantageously improves user noticeable delays and slowdowns due to the lower transmission rate. The extension of data requests from the user network 510 beyond the access server 524 is eliminated when the data requested is cached in the database system. Also, using a predictive algorithm reduces delay by improving caching efficiency based on a user's predictable pattern.

FIG. 40 depicts a flowchart for the access server 524 for auditing in an example of the invention. FIG. 40 begins at step 4000. In step 4002, the user network 510 establishes a connection to the access server 524. In step 4004, the access server 524 generates and transmits a login request message to the database system 522. The access server 524 then checks if the database system allowed an access session to be established for the user in step 4006.

If the access session is not established, the access server 524 terminates in step 4020. If the access session is established, the access server 524 then checks if an access session tear down is requested in step 4012. If the access session tear down is requested, the access server 524 generates and transmits a tear down message with user and path information to the database system 522 in step 4014 before terminating in step 4020.

While no session tear down request is received, the access server 524 exchanges packets with the user network 510 depending on the service provided in step 4016. The access server 524 transmits all new packet destinations including the user and path information to the database system 522 in step 4018 before returning to step 4012. In one embodiment, the database system 522 stores the connect request message, the session reject message, the session accept message, the tear down message including the user and path information, and the packets including the user and path information in an audit database system 526. In one embodiment, the connect request message, the session reject message, the session accept message, the tear down message including the user and path information, and the packets including the user and path information are in the form of an audit message. An audit message is any message, information, or signaling that is audited while a user is accessing an access network 520.

FIG. 41 depicts a flowchart for the database system 522 for auditing in user predictive caching in an example of the invention. FIG. 41 begins in step 4100. In step 4102, the database system 522 waits for an audit message. In step 4104, the database system 522 receives and processes the audit message. The database system 522 then checks if the audit message is a session accept message or a tear down message in step 4106. If the audit message is not a session accept message or a tear down message, the database system 522 retrieves the session state object and updates the object with the audit message in step 4112 before returning to step 4102.

If the audit message is a session accept message or a tear down message, the database system 522 then checks if the audit message is a session accept message in step 4108. If the audit message is a session accept message, the database system 522 generates a session state object including the user, device, path, and session ID in step 4114 before returning to step 1802. If the audit message is a tear down message, the database system 522 retrieves the session state object and updates the object in step 4112. Also, the database system 522 stores the event in step 4116. In one embodiment, the database system 522 stores the event in the cache database system 574. The database system 522 removes the active reference from the session state object before returning to step 4102. In one embodiment of the invention, the database system 522 uses the audit database system 573 to perform the operations disclosed in FIG. 41.

FIG. 42 depicts a flowchart for the database system 522 for caching in user predictive caching in an example of the invention. FIG. 42 begins in step 4200. In step 4202, the database system 522 waits for the next event. The database system 522 processes the event in step 4204. In this embodiment, a request for data is a cache request. The database system 522 then checks if the event is a cache request in step 4206.

If the event is a cache request, the database system 522 identifies the user and retrieves the cached user data in step 4208. The database system 522 then checks if the requested data from the cache request is in the cached user data in step 4212. If the requested data is in the cached user data, the database system 522 replies with the requested data in step 4210 before returning to step 4202. If the requested data is not in the cached user data, the database system 522 retrieves the requested data from the service provider via the access server 524 in step 4214. The database system 524 then replies to the user network 510 with the requested data. Alternatively, in another embodiment, the access server 524 transfers the requested data from the service provider to the user network 510 and the database system 522.

In step 4222, the database system 522 updates the user access profile and the caching algorithm. The database system 522 then determines whether there is a change in the cached data based on the caching algorithm in step 4230. If there is no change in the cached data, the database system 522 returns to step 4202. If there is a change in the cached data, the database system 522 sets any new caching timers for the cached data in step 4236. The database system 522 also generates and transmits a translation change to the access server 524 to change the requested destination address to the database system 522 in step 4238 before returning to step 4202.

If the event is not a cache request, the database system 522 then checks if the event is an audit or timer request in step 4216. If the event is an audit or timer request, the database system 522 identifies the user and retrieves the user access profile in step 4218. The database system 522 then checks if the event is an audit request in step 4220. If the event is an audit request, the database system 522 proceeds to step 4222. If the event is not an audit request, the database system 522 refreshes the cached data from the service provider of the cached data based on the caching algorithm in step 4228. The database system 522 then resets the caching timer for the cached data in step 4234 before returning to step 4202.

If the event is not an audit or timer request, the database system 522 checks if the event is a reset event in step 4224. If the event is not a reset event, the database system 522 registers a unknown event received in step 4232 before returning to step 4202. If the event is a reset event, the database system 522 clears the user access profile statistics, caching timers, the user's cached data. In one embodiment, the user's cached data is stored in the cached database system 574 and the database system 522 uses the cached database system 574 to perform the operations disclosed in FIG. 42.

User Controlled Caching

Another solution to reduce user noticeable delays and slowdowns due to the lower transmission rate is user controlled caching. FIGS. 5, 40, 43 and 44 disclose one embodiment of the invention for user controlled caching. A user controls caching by creating a script of commands to cache data prior to the user accessing the network. For example, a user selects financial or local weather forecasts to cache, so when the user logs on to the network the information will be immediately available from the cache. Scripting has typi-

cally been done on general purpose computers. In a general purpose computer, the user sets up a sequenced set of commands to be executed when the script is executed. Unfortunately, this type of scripting has not been performed on a network cache.

In this embodiment of user controlled caching, the operation of the access server 524 is as described in FIG. 40. FIG. 43 depicts a flowchart for the database system 522 for auditing in user controlled caching in an example of the invention. FIG. 43 begins in step 4300. In step 4302, the database system 522 waits for an audit message. In step 4302, the database system 522 receives and processes the audit message. The database system 522 then checks if the audit message is a session accept message or a tear down message in step 4306. If the audit message is not a session accept message or a tear down message, the database system 522 retrieves the session state object and updates the object with the audit message in the cache in step 4312 before returning to step 4302.

If the audit message is a session accept message or a tear down message, the database system 522 then checks if the audit message is a session accept message in step 4308. If the audit message is a session accept message, the database system 522 generates a session state object in the cache including the user, device, path, and session ID in step 4314. The database system 522 then checks if the user has a pre-cache script in step 4318. If the user does not have a pre-cache script, the database system 522 returns to step 4302. If the user has a pre-cache script, the database system 522 generates and transmits a pre-cache instruction set to the access server 524 to change the translation filter to access the database system 522 for destinations in the pre-cache script so that any requests for those destinations are fulfilled from the cache in step 4320. The database system 522 then sets cache refresh timers in step 4322 and returns to step 4302.

If the audit message is a tear down message, the database system 522 retrieves the session state object and updates the object in step 4316. Also, the database system 522 stores the event in step 4316. In one embodiment, the database system 522 stores the session state object and the state in the cache database system 527. The database system 522 removes the active reference from the session state object before returning to step 4302. In one embodiment of the invention, the database system 522 uses the audit database system 573 to perform the operations disclosed in FIG. 43.

FIG. 44 depicts a flowchart for the database system 522 for caching in user controlled caching in an example of the invention. FIG. 44 begins in step 4400. In step 4402, the database system 522 waits for the next event. The database system 522 processes the event in step 4404. In this embodiment, a request for data is a cache request. The database system 522 then checks if the event is a cache request in step 4406.

If the event is a cache request, the database system 522 identifies the user and retrieves the cached user data in step 4408. The database system 522 then checks if the requested data from the cache request is in the cached user data in step 4412. If the requested data is in the cached user data, the database system 522 replies with the requested data in step 4410 before returning to step 4402. If the requested data is not in the cached user data, the database system 522 retrieves the requested data from the service provider via the access server 524 in step 4414. The database system 522 then replies to the user network 510 with the requested data. Alternatively, in another embodiment, the access server 524 transfers the requested data from the service provider to the user network 510 and the database system 522.

In step 4422, the database system 522 updates the user access profile and the caching algorithm. The database system 522 then determines whether there is a change in the cached data based on the caching algorithm in step 4430. If there is no change in the cached data, the database system 522 returns to step 4402. If there is a change in the cached data, the database system 522 sets the new caching timer for the cached data in step 4436. The database system 522 also generates and transmits a translation change to the access server 524 to change the requested destination address to the database system 522 in step 4440 before returning to step 4402.

If the event is not a cache request, the database system 522 then checks if the event is an audit or timer request in step 4416. If the event is an audit or timer request, the database system 522 identifies the user and retrieves the user access profile in step 4418. The database system 522 then checks if the event is an audit request in step 4420. If the event is an audit request, the database system 522 proceeds to step 4422.

If the event is not an audit request, the database system 522 checks if the event is a script timer in step 4428. If the event is not a script timer, the database system 522 refreshes the cached data from the service provider of the cached data based on the caching algorithm in step 4438 before returning to step 4402. If the event is a script timer, the database system 522 executes the user commands or command set specified by the script timer event in step 4434. The database system 522 then resets the script timer for the cached data in step 4442 before returning to step 4402.

If the event is not an audit or timer request, the database system 522 checks if the event is a reset event in step 4424. If the event is not a reset event, the database system 522 registers a unknown event received in step 4432 before returning to step 4402. If the event is a reset event, the database system 522 clears the user access profile statistics, caching timers, the user's cached data. In one embodiment, the user's cached data is stored in the cached database system 574 and the database system 522 uses the cached database system 574 to perform the operations disclosed in FIG. 44.

Access Network Switching— FIGS. 5, 45, 46, and 47

Switching Access by a User

Access between users and service providers vary based on quality and security, which in turn determine the costs of the access. Users typically have a need to switch between types of access depending on the service offered or what stage in the service the user is in. For example, a user browses the amazon.com website for books using a standard Internet access. When purchasing the books, the user needs a more secure Internet access to ensure that the user's credit card number is not stolen by a hacker.

One prior solution for enhanced security is data Virtual Private Networks (VPN). Data VPN's are relatively static constructions which allow the extension of a user network to another location by extending the network over leased lines or shared Internet/Intranet facilities. However, VPN's require service anticipation, planning and expense beyond what the typical Internet and Intranet users possess. VPN's also do not provide dynamic switching between accesses. Another prior solution is Local Area Network emulation (LANE). LANE utilizes ATM transports to extend the reach of the user network. However, most Internet and Intranet users do not possess ATM equipment and expertise, and the backbone network is still IP.

FIGS. 5 and 45 disclose one embodiment for switching access by a user in an example of the invention. A user

selects a different access to a service provider and is switched to the access without the user's connection to their access provider being interrupted. FIG. 45 depicts a flowchart for switching access by a user in an example of the invention. In this embodiment, a user using standard Internet access selects a premiere Internet access during the setup of a service application. There are numerous variations in switching between access paths such as from premiere Internet access path to a lower quality Internet access path, but only one embodiment is shown for the sake of simplicity.

FIG. 45 begins in step 4500. In step 4502, a user through the network device 512 transmits a request using a standard Internet access path through the service network 530 for a service application residing in the network device 562. The service application in the network device 562 receives the request and transmits a query for the quality of service (QOS) Internet access desired by the requester to the network device 512 in step 4504. The service application in the network device 562 then receives and processes a request to switch access to check if the user selected a premium QOS Internet access in step 4516. If the user did not select a premium QOS Internet access, the user and service application exchange packets using the standard Internet access via the network device 512, the access server 524, the service network 530, the access server 554, and the network device 562 in step 4516. The service application in the network device 562 returns the control to the user in step 4514 to select another service application in step 4502.

If the user selects a premium QOS Internet access, the service application in the network device 562 generates and transmits an authentication and authorization instruction for the premium QOS Internet access to the database system 522 to check if the switch of access is allowed. The database system 522 receives and processes the authentication and authorization instruction. The database system 522 then generates and transmits a premium access instruction to establish premium Internet access between the access server 524 and the access server 554 via the service network 540. In one embodiment, the database system 522 transmits the premium access instruction to the service network 540 to establish premium Internet access between the access server 524 and the access server 554. The database system 522 then generates and transmits the premium access instruction to the access server 524 to connect the network device 512 to the service network 540 for the premium Internet access. The database system 522 also generates and transmits the premium access instruction to the access server 554 to connect the network device 562 to the service network 540 for the premium Internet access.

Once the premium Internet access is established, the user and service application exchange packets using the premium Internet access via the network device 512, the access server 524, the service network 540, the access server 554, and the network device 562 in step 4512. The database system 522 then generates and transmits a disconnect instruction to the access server 524, the service network 540, and the access server 554 to tear down the premium Internet access in step 4518. The database system 522 then generates and transmits usage and performance statistics for the service application in the network device 562 in step 4520 before returning to step 4514. In one embodiment, the database system 522 uses a cross connect system 583 to perform the operations disclosed in FIG. 45.

Switching Access by a Service Provider

FIGS. 5 and 46 disclose one embodiment for switching access by a service provider in an example of the invention. The service provider selects a different access to a user and

41

is switched to the access without the user or service provider's connection to their access provider being interrupted. FIG. 46 depicts a flowchart for switching access by a service provider in an example of the invention. In this embodiment, the service provider selects a toll free premium Internet service for a user using standard Internet access. The service provider pays for the changes in the toll free premium Internet service instead of the user—similar to toll free phone calls.

FIG. 46 begins in step 4600. In step 4602, the service application in the network device 562 waits for a request for a service. The user through the network device 512 transmits a request using a standard Internet access through the service network 530 for the service application in the network device 562 in step 4604. The service application in the network device 562 receives and processes the request. The service application then selects a toll free premium quality of service (QOS) Internet access to the network device 512 in step 4606. The service application in the network device 562 generates and transmits an authentication instruction to the database system 522 in step 4608. The database system 522 receives and processes the authentication instruction. In step 4612, the database system 522 then generates and transmits a premium access instruction to establish the toll free premium Internet access between the access server 524 and the access server 554 via the service network 540.

Once the toll free premium Internet access is established, the user and service application exchange packets using the premium Internet access via the network device 512, the access server 524, the service network 540, the access server 554, and the network device 562 in step 4612. Once the service is completed, the database system 522 then generates and transmits a disconnect instruction to the access server 524, the service network 540, and the access server 554 to tear down the toll free premium Internet access in step 4614. The database system 522 then generates and transmits usage and performance statistics for the service application in the network device 562 in step 4616. Once the service is completed, the service application in the network device 562 returns the control to the user in step 4618 to select another service application in step 4602. In one embodiment, the database system 522 uses the cross connection system 583 to perform the operations disclosed in FIG. 46. In another embodiment, a third party such as the government performs the switching for surveillance or monitoring purposes.

Dynamic Switching Access

FIGS. 5 and 47 disclose one embodiment for dynamic switching access in an example of the invention. In a prior solution, a single access link from the user network to the access server is linked to a dedicated network in a single access session. However, users need to switch between different networks such as Internet and Intranets without tearing down the existing access session. No other systems allow the access server to be controlled by the data stream the access server is passing. Analog modems monitor the character stream for an escape sequence. The escape sequence notifies the modem that the data is for modem control and not for application data. However, no prior systems have implemented this functionality for a packet data network.

FIG. 47 depicts a flowchart for dynamic switching access in an example of the invention. In this embodiment, a user during an access session may switch networks without having the access session torn down. The access server 524 receives a control instruction from the user to switch from one dedicated service network 530 to another dedicated service network 540. FIG. 47 begins in step 4700. In step

42

4702, the access server 524 waits for a packet. In this embodiment, a request is in the format of a packet. The access server 524 then receives and processes the packet from the network device 512 in step 4704. The access server 524 checks if the packet is encoded for access control in step 4706. If the packet is not encoded for access control, the access server 524 processes the packet using standard access logic in step 4714 before returning to step 4702.

If the packet is encoded for access control, the access server 524 identifies the physical access path characteristics in step 4708. The access server 524 then checks if the access control is allowed in step 4710. If access control is not allowed, the access server 524 registers and logs an illegal access event in step 4716. The access server 524 then discards the packet in step 4718 before returning to step 4702.

If access control is allowed, the access server 524 generates and transmits an access control instruction to the database system 522 in step 4712. The database system 522 then receives and processes the access control instruction. The database system 522 identifies, authenticates, and authorizes the user and the requesting access server using the packet and path in step 4720. The database system 522 then retrieves the user access profile and the network device profile in step 4722. The database system 522 then checks if access control is allowed for the user and the network device based on the user access profile and the network device profile in step 4724. If access is not allowed, the database system 522 proceeds to step 4716.

If access is allowed, the database system 522 generates and transmits an access instruction to the access server 524 to update the access logic to switch to the service network 540 in step 4726. The database system 522 logs the access change in step 4727. In one embodiment, the database system 522 logs the access change in the audit system 572. The database system 522 also generates and transmits a reply with the new access status to the network device 512 in step 4728 before returning to step 4702. In one embodiment, the database system 522 uses the cross connect system 583 to perform the operations disclosed in FIG. 47.

Access Network Destination Control—FIGS. 5, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, and 59

Network devices become unavailable for a various reasons such as busy, failure, or overload. For network failover, when a destination network device is unavailable, network device requests are manually re-routed to a secondary network device. In order to avoid this manual rerouting, one prior art solution is the sharing of a network address between multiple processors in a box or multiple machines in a cluster. However, the requirement of destination and secondary devices residing in the same box or cluster makes this solution unacceptable for large systems.

Another prior art system is Network Address Translation, which translates a private network address to a public network address for a period of time. This system does not provide the capability of a network address to represent multiple failover destinations. Another solution is the Common Object Request Broker Architecture (CORBA), which provides the user a list of potential "logical" network devices to post the network device request. However, the user must implement CORBA interfaces and interactively select back up processes. The transaction context is lost when the primary fails. The user must re-establish the context. Unfortunately, none of these solutions provide an automatic translation from an unavailable destination network device to a secondary network device without operator

or user intervention, where the destination and secondary devices are not within the same box or cluster.

FIGS. 5, 48 and 49 disclose one embodiment for handling network address failover in an example of the invention. In a failover scenario, a destination network device fails or overloads, which makes the destination network device unavailable, and a secondary network device replicates the destination network device to give the user a fault tolerant appearance of the destination network device. Requests for the destination network device are translated to the secondary network device.

FIG. 48 depicts a flowchart for the access server 554 for network address failover in an example of the invention. FIG. 48 begins in step 4800. In step 4802, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 4804. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 4806, the access server 554 detects that the destination network device 562 fails and checks if the destination network address is enabled for failover in step 4806. If the destination network address is not enabled for failover, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 4812 before returning to step 4802.

If the destination network address is enabled for failover, the access server 554 checks if an address translation is active for the destination network device 562 in step 4808. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 4812 before returning to step 4802. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for failover in step 4810. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 4812 before returning to step 4802.

FIG. 49 depicts a flowchart for the database system 522 for network address failover in an example of the invention. FIG. 49 begins in step 4900. In step 4902, the database system 522 waits for a state event message or a timer event from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or a network device timer expires in step 4904. The database system 522 checks if the network device timer expired in step 4906.

If the network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout in step 4908. The database system 522 then retrieves and updates and the secondary network device's 564 profile to indicate that the destination network device 562 is unavailable in step 4910. In step 4912, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer has not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 4914. In step 4916, the database system 522 checks if the destination network device 562 is known. If the destination network device 562 is not known, the database system 522 generates

and transmits an unknown resource event before returning to step 4902. If the destination network device 562 is known, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics in step 4918. The database system 522 also resets the network device timer. The database system 522 then checks if the state event message is a failure or start event in step 4920.

If the state event message is not a failure or start event, the database system 522 checks if the load event threshold for the destination network device 562 is reached in step 4930. If the load event threshold is not reached, the database system 522 returns to step 4902. If the load event threshold is reached, the database system 522 proceeds to step 4926.

If the state event message is a failure or start event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 4922. In step 4924, the database system 522 checks if the state event message is a failure event. If the state event message is not a failure event, the database system 522 generates new active device profiles and then returns to step 4902. If the state event message is a failure event, the database system 522 proceeds to step 4926.

In step 4926, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address in step 4932 before returning to step 4902. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event in step 4928 before returning to step 4902. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 49.

Network Busy Forwarding

In a Public Switched Telephone Network (PSTN), a busy signal is sent to the caller when the called number destination is busy and thus unavailable. The caller can then select alternate actions such as delaying, retrying, or re-routing the request. Unfortunately, the PSTN handling of busy calls has not been applied to network devices in a data network. FIGS. 5, 50 and 51 disclose one embodiment for network busy forwarding in an example of the invention. When a destination network device 562 is busy, an access server 554 forwards the packets to a secondary network device 564. Requests for the destination network device are translated to the secondary network device.

FIG. 50 depicts a flowchart for the access server 554 for network busy forwarding in an example of the invention. FIG. 50 begins in step 5000. In step 5002, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5004. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5006, the access server 554 checks if the destination network device 562 is busy. If the destination network device 562 is not busy, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5012 before returning to step 5002.

If the destination network device 562 is busy, the access server 554 checks if an address translation is active for the destination network device 562 in step 5008. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step

5012 before returning to step 5002. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for failover in step 5010. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 5012 before returning to step 5002.

FIG. 51 depicts a flowchart for the database system 522 for network busy forwarding in an example of the invention. FIG. 51 begins in step 5100. In step 5102, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or the network device timer expired in step 5104. The database system 522 checks if a network device timer expired in step 5106.

If a network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout such as the network device being busy or unavailable in step 5108. The database system 522 then retrieves and updates and the secondary network device's 564 profile to indicate that the destination network device 562 is busy or unavailable in step 5110. The database system 522 also resets the network device timer. In step 5112, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer is not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 5114. In step 5116, the database system 522 checks if the destination network device 562 is known. If the destination network device 562 is not known, the database system 522 generates and transmits an unknown resource event before returning to step 5102. If the destination network device 562 is known, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics in step 5118. The database system 522 then checks if the state event message is a failure/busy or start/available event in step 5120.

If the state event message is not a failure/busy or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5130. If the load event threshold is not reached, the database system 522 returns to step 5102. If the load event threshold is reached, the database system 522 proceeds to step 5126.

If the state event message is a failure/busy or start/available event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 5122. In step 5124, the database system 522 checks if the state event message is a failure/busy event. If the state event message is not a failure/busy event, the database system 522 generates an address translation clear request in step 5132 before returning to step 5102. The address translation clear request adds a new active device profile or clears the existing translation. If the state event message is a failure/busy event, the database system 522 proceeds to step 5126.

In step 5126, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address if a busy

translation exists in the destination network device's 562 profile in step 5134 before returning to step 5102. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5128 before returning to step 5102. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 51.

Busy Flag

FIGS. 5 and 52 disclose one embodiment for a busy flag when the destination network device is busy in an example of the invention. When a destination network device 562 is busy and thus unavailable, an access server 554 replies to the user with a busy flag. FIG. 52 depicts a flowchart for the access server 554 for a busy flag when the destination network device is busy in an example of the invention. In one embodiment, the busy flag is a busy screen in HyperText Markup Language. FIG. 52 begins in step 5200. In step 5204, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5204. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5206, the access server 554 checks if the destination network device 562 is busy. If the destination network device 562 is not busy, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5210 before returning to step 5202. If the destination network device 562 is busy, the access server 554 replies to the user with a busy flag in step 5208 before returning to step 5202.

Timeout

FIGS. 5 and 53 disclose one embodiment for forwarding if the destination network device times out in an example of the invention. In this embodiment, a destination network device 562 receives a request packet from a user but fails to reply to the user within a pre-determined time. The failure to reply within a pre-determined time indicates the destination network device 562 is unavailable. The access server 554 then redirects the packet to a secondary network device 564 to handle forwarding when the destination network device fails to respond with the pre-determined time.

FIG. 53 depicts a flowchart for the access server 554 for forwarding if the destination network device times out in an example of the invention. FIG. 53 begins in step 5300. In step 5302, the access server 554 waits for the next packet or event. The access server 554 then checks whether a reply timeout event is received in step 5304. If a reply timeout event is not received, the access server 554 then receives and processes the packet in step 5306. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5308, the access server 554 checks if an address translation is available for the destination network device 562. If an address translation is not available for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5318 before returning to step 5302. If an address translation is available for the destination network device 562, the access server 554 sets a reply timer and caches the packet for the secondary network device 564. The access server 554 then transmits the packet with standard access logic to the destination network device 564 in step 5318 before returning to step 5302.

In step 5312, the access server 554 receives the reply timeout event and retrieves the cached request. The access server 554 then processes the reply timeout event and the cached request in step 5314. In step 5315, the access server

554 then replaces the packet's destination network address with the cached secondary network address. The access server 554 then transmits the packet to the secondary network device 564 with the standard access logic in step 5318 before returning to step 5302. In one embodiment, upon reply, the access server 554 deletes the cached packet copy.

Scheduled Alias Resolution

Network devices become unavailable for various reasons such as busy, failure, or overload. Also, service provider need to utilize network resources to improve efficiency and avoid network device latency. In voice telephony signaling networks, an SS7 SCP or an automatic call distributor (ACD) distributes calls to end user call centers. The SCP receives a 800 TCAP query and performs a database lookup to translate the 800 telephone number into a destination telephone number. The SCP may use the requester information and the time of day to perform the database lookup. Some problems with this system are SS7 signaling must be used and the system is limited by voice network constraints. Also, the phone numbers only allow numeric numbers for aliases. In data networks, prior art known as Object Request Brokers (ORB) in a Common Object Request Broker Architecture (CORBA) advertise process resources to a requester. The ORB also registers multiple instances of the same type of process resource.

FIGS. 5, 54, and 55 disclose one embodiment for scheduled alias resolution in an example of the invention. In this embodiment, an alias, domain name/local name, network address, or macro for a network resource is translated to another alias for another network resource based on a configurable schedule. Advantageously, load balancing of network devices is improved by scheduling different alias resolutions for different times/conditions. One problem with CORBA is the requester and resources must implement CORBA interfaces. Scheduled alias resolution is compatible with existing data network DNS implementations. Also, the load balancing, security, and failover may be based on the user access profile and the network device state.

FIG. 54 depicts a flowchart for the access server 554 for schedule alias resolution in an example of the invention. FIG. 54 begins in step 5400. In step 5402, the access server 554 waits for the next request. The access server 554 then receives and processes a request from the network device 512 in step 5404. In this embodiment, the information including an alias is in the form of a request. In step 5406, the access server 554 checks if a current translation for the alias exists. If a current translation for the alias exists, the access server 554 translates the alias to the appropriate network address for a network device in user network 560 in step 5408. The access server 554 then replies to the network device 512 in step 5410 before returning to step 5402.

If a current translation for the alias does not exist, the access server 554 checks if a known access profile exists in the database system 522 exists for this user in step 5412. If a known access profile does not exist in the database system 522 for the user, the access server 554 generates and transmits an alias resolution failure message to the network device 512 in step 5414 before returning to step 5402. If a known access profile does exist in the database system 522 for the user, the access server 554 sets an external request timer in step 5416. The access server 554 then generates and transmits an alias translation request to the database system 522 in step 5418. The access server 554 then checks if a reply is received or the external request timer is expired in step 5420. If the reply is not received or the external request timer is not expired, the access server 554 returns to step

5420. If the reply is received or the external request timer is expired, the access server 554 checks if the reply was valid in step 5422. If the reply is invalid, the access server 554 returns to step 5414. If the reply is valid, the access server 554 then caches the alias translation for this user with time to live parameters in step 5424 before returning to step 5408.

FIG. 55 depicts a flow chart for the database system 522 for scheduled alias resolution in an example of the invention. FIG. 55 begins in step 5500. In step 5502, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 554 in step 5504. In step 5506, the database system 522 then checks if the access server requester is known in step 5506. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 5508. The database system 522 appends the reply to the packet and transmits the reply to the access server 554 in step 5510. The database system 522 replies with a decline message to the access server 554 in step 5512 before returning to step 5502.

If the access server requester is known, the database system 522 then checks whether the request is a translation appeal in step 5514. If the request is not a translation appeal, the database system 522 checks if the request is a security event in step 5516. If the request is a security event, the database system 522 increments a path/device security record in step 5522 before returning to step 5512. If the request is not a security event, the database system 522 registers an unknown event in step 5530 before returning to step 5502. If the request is a translation appeal, the database system 522 identifies and authenticates the user and the network device 512 in step 5518. If the access is not valid, the database system 522 returns to step 5522. If the access is valid, the database system 522 retrieves the user and network device's 512 profiles in step 5524. The database system 522 then generates the translation logic and calculates the translation time to live in step 5526. The database system 522 then appends the translation logic and the translation time to a reply to the access server 554. The database system 522 then replies with an approve message to the access server 554 in step 5528.

Forwarding

In a PSTN, a person may forward calls from their original phone number to another phone number where the person may be reached. Unfortunately, the PSTN handling of call forwarding has not been applied to network devices in a data network. FIGS. 5, 56 and 57 disclose one embodiment for destination controlled forwarding in an example of the invention. An access server 554 forwards the packets for a destination network device 562 that is unavailable to a secondary network device 564. Requests for the destination network device 562 are translated to the secondary network device 564.

FIG. 56 depicts a flowchart for the access server 554 for destination controlled forwarding in an example of the invention. FIG. 56 begins in step 5600. In step 5602, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5604. In this embodiment, the access server 554 receives and processes information in the form of a packet. The access server 554 then checks if the packet is a reply in step 5606. If the packet is a reply, the access server 554 retrieves and deletes the cached destination network address in step 5608 before proceeding to step 5614.

If the packet is not a reply, the access server 554 checks if an address translation is active for the destination network

device 562 in step 5610. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5614 before returning to step 5602. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for destination controlled forwarding in step 5612. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 5614 before returning to step 5602.

FIG. 57 depicts a flowchart for the database system 522 for destination controlled forwarding in an example of the invention. FIG. 57 begins in step 5700. In step 5702, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or a network device timer expires in step 5704.

The database system 522 processes the state event message and authenticates the destination network device 562 in step 5706. In step 5708, the database system 522 checks if the access server requester is known. If the access server requester is not known, the database system 522 generates and transmits an unknown requester event before returning to step 5702. If the access server requester is known, the database system 522 retrieves and updates the access server's 554 profile with availability and performance statistics in step 5710. The database system 522 then checks if the state event message is a failure/forward or start/available event in step 5712.

If the state event message is not a failure/forward or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5716. If the load event threshold is not reached, the database system 522 returns to step 5702. If the load event threshold is reached, the database system 522 proceeds to step 5722.

If the state event message is a failure/forward or start/available event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 5718. In step 5720, the database system 522 checks if the state event message is a failure/forward event. If the state event message is not a failure/forward event, the database system 522 generates and transmits an address translation clear request message to the access server 554 in step 5726 before returning to step 5702. If the state event is a start event, then a new availability profile is generated. If the state event message is a failure/busy event, the database system 522 proceeds to step 5722.

In step 5722, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address if a busy translation exists in the destination network device's 562 profile in step 5728 before returning to step 5702. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5724 before returning to step 5702. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 57.

Network Endpoint Availability Management

FIGS. 5, 58, and 59 disclose one embodiment for network endpoint availability management in an example of the invention. The features of failover, busy, busy flag, timeout, and forwarding are combined into this embodiment. FIG. 58 depicts a flowchart for the access server 554 network endpoint availability management in an example of the invention. FIG. 58 begins in step 5800. In step 5802, the access server 554 waits for the next packet or event. The access server 554 then checks whether a reply timeout event is received in step 5804.

If a reply timeout event is not received, the access server 554 then receives a data packet and processes the packet in step 5806. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5808, the access server 554 checks if the packet is a reply. If the packet is a reply, the access server 554 clears the cached request and clears a reply timer in step 5810 before returning to step 5804. If the packet is not a reply, the access server 554 checks if an address translation is active for the destination network device 562 in step 5812. If an address translation is active for the destination network device 562, the access server 554 proceeds to step 5824. If an address translation is not active for the destination network device 562, the access server 554 checks if the destination address is busy. If the destination address is not busy, the access server 554 proceeds to step 5830. If the destination address is not busy, the access server 554 checks if a forward address translation is available in step 5814. If the forward address translation is not available, replies to the user with a busy flag in step 5818 before returning to step 5802. If the forward address translation is available, the access server 554 proceeds to step 5824.

If a reply timeout event is received, the access server 554 receives the reply timeout event and the cached request in step 5820. The access server 554 then processes the reply timeout event and the cached request in step 5822. In step 5824, the access server 554 then replaces the packet's destination network address with the cached secondary network address. The access server 554 then transmits the packet to the secondary network device 564 with the standard access logic in step 5824.

In step 5826, the access server 554 checks if the forward address is busy. If the forward address is busy, the access server 554 returns to step 5818. If the forward address is not busy, the access server 554 sets a reply timer and caches the request in step 5828. In step 5830, the access server 554 then transmits the packet with standard access logic before returning to step 5802.

FIG. 59 depicts a flowchart for the database system 522 for network endpoint availability management in an example of the invention. FIG. 59 begins in step 5900. In step 5902, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives the state event message from the destination network device 562 or the network device timer expired in step 5904. The database system 522 checks if the network device timer expired in step 5906.

If the network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout such as the network device being busy or unavailable in step 5908. The database system 522 then retrieves and updates the secondary network device's 564 profile to indicate that the destination network device 562 is busy or unavailable in step 5910. The database system 522 also resets the network device timer. In step 5912, the database system 522 generates and transmits an

51

address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer has not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 5914. In step 5916, the database system 522 checks if the access server requester is known. If the access server requester is not known, the database system 522 generates and transmits an unknown resource event before returning to step 5902. If the access server requester is known, the database system 522 retrieves and updates the access server's 554 profile with availability, activity logs, and performance statistics in step 5918. The database system 522 then checks if the state event message is a failure, busy, forward or start/available event in step 5920.

If the state event message is not a failure, busy, forward or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5930. If the load event threshold is not reached, the database system 522 returns to step 5102. If the load event threshold is reached, the database system 522 proceeds to step 5926.

If the state event message is a failure, busy, forward, or start/available event, the database system 522 retrieves and updates the destination network device 562 and secondary network device's 564 profile in step 5922. In step 5124, the database system 522 checks if the state event message is a failure, busy, or forward event. If the state event message is not a failure/busy event, the database system 522 generates and transmits an address translation clear request to the access server 554 if a busy translation exists in step 5934 before returning to step 5902. If the state event message is a failure, busy, or forward event, the database system 522 proceeds to step 5926.

In step 5926, the database system 522 checks if a secondary network device 564 is available for forwarding. If a secondary network device 564 is available, the database system 522 checks if the state event message is a failure or busy event in step 5936. If the state event message is a failure or busy event, the database system 522 generates and transmits a busy address translation request message to the access server 554 in step 5932 before returning to step 5902. If the state event message is not a failure or busy event, the database system 522 generates and transmits a forward address translation request message to the access server 554 to change packets' destination network address to the secondary network address in step 5938 before returning to step 5902. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5928 before returning to step 5902. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 59.

Access Network Audit Server—

FIGS. 5, 60, 61, 62, and 63

Service Capability Monitor

FIGS. 5, 60, 61, 62, and 63 disclose one embodiment for service capability monitor in an example of the invention. In a prior system, Java applet technology and Sun's JINI project delivers communication and state mechanisms from the function provider to the function requesters. Java applet technology allows a mechanism for device control without any coordinating control agent. JINI provides a registry which maintains capabilities like a CORBA ORB as a means for primitive control. One problem is that JINI focuses on

52

the distribution of mechanisms instead of the interworking of mechanisms in a standardized way. In this embodiment of service capability control, a network access provider monitors the validity and state of the application level request to a network service provider. The network access provider monitors performance of their network and the service provider's network service requiring an understanding of the other provider's service requested. Access providers such as Internet service providers could mitigate their liability and increase the service's performance by extending their service view to the user base. Specific service environment access points may be monitored and service access rights and restrictions may be enforced. This embodiment provides an interworking mechanism that provides a dynamic interface to a service based network. The monitoring interface must be dynamic because different service have different potential states, different legal and illegal state transitions, and different communication mechanisms.

FIG. 60 depicts a flowchart for a firewall 526 for service capability monitor in an example of the invention. FIG. 60 begins in step 6000. In step 6002, the firewall 526 waits for a packet. The firewall 526 then receives and processes a packet from the network device 512 in step 6004. In step 6006, the firewall 526 performs standard firewall functions. The firewall 526 then copies the packets and transmits the packet to the destination and the database system 522 in step 6008 before returning step 6002.

FIG. 61 depicts a service capability monitor software architecture for a service capability monitor in an example of the invention. A service capability monitor software architecture 6100 comprises a service broker tokenizer process B 6112, a service broker state map process C 6114, a network A controller process A 6116, a service A signal tokenizer process E 6122, a service A state map process F 6124, a service A controller process D 6126, a service B signal tokenizer process H 6132, a service B state map process F 6134, and a service B controller process D 6136. The network A controller process A 6116 includes a service broker session state objects 6118. The service A controller process D 6126 includes a service A session state objects 6128. The service B controller process D 6136 includes a service B session state object 6138.

The network A controller process A 6116 is connected to the service broker tokenizer process B 6112, the service broker state map process C 6114, the service A controller process D 6126, and the service B controller process D 6136. The service A controller process D 6126 is connected to the service A signal tokenizer process E 6122 and the service A state map process F 6124. The service B controller process D 6136 is connected to the service B signal tokenizer process H 6132 and the service B state map process F 6134.

FIG. 62 depicts a flowchart for the network logic for service capability monitor in an example of the invention. FIG. 62 begins in step 6200. In step 6202, the database system 522 waits for the packet. The database system 522 then receives and processes the packet from the firewall 526 in step 6204. The database system 522 then identifies and authorizes the requester in step 6206. The database system 522 checks if the requester is known in step 6208.

If the requester is known, the network A controller process 6116 writes the packet to process B—service broker signal tokenizer process 6112 in step 6210. The process B 6112 then replies with the function token and parameters to the process A 6116 in step 6212. The database system 522 then checks if a service session state object (SSO) 6118 exists for this session in step 6214. If the SSO 6118 does not exist, the database system 522 creates a SSO 6118 for this session and

53

initializes the current state in step 6216. If the SSO 6118 exists, the database system 522 then proceeds to step 6218. In step 6218, process A 6116 writes the state token and current state to process C—service broker state map process 6114 in step 6218. The process C 6114 then replies with a transition token and parameters to the process A 6116 in step 6220. The process A 6116 applies the transition token and the parameters to the SSO 6118 to update the current state and setting actions in step 6222.

In step 6224, the database system 522 checks if there is an action indicated by the SSO 6118. If there is not an action indicated by the SSO 6118, the database system 522 proceeds to step 6228. If there is an action indicated by the SSO 6118, the database system 522 performs the transition action specified by the SSO 6118 in step 6226. In step 6228, the database system 522 updates the session sliding usage accumulators. The database system 522 then checks if there is an action in the SSO 6118 for this usage in step 6230. If there is no action, the database system 522 returns to step 6202. If there is an action in the SSO 6118, the database system 522 performs the usage action specified in the SSO 6118 before returning to step 6202.

If the requester is not known, the database system 522 checks if there is a network A state object in step 6234. If there is a network state object, the database system 522 proceeds to step 6238. If there is no network state object, the database system 522 creates a network A state object and initializes the current state in step 6236. In step 6238, the database system 522 increments network A state object usage for this event. The database system 522 then checks if there is an action in the network A state object for this usage in step 6240. If there is no action in the network A state object for this usage, the database system 522 returns to step 6202. If there is an action in the network A state object for this usage, the database system 522 performs the usage action specified in the network A state object in step 6242 before returning to step 6242.

FIG. 63 depicts a flowchart for the service logic for service capability monitor in an example of the invention. FIG. 63 begins in step 6300. In step 6302, process D service A controller process 6126 waits for the packet. The process D 6126 then receives and processes the packet from the process A network A controller 6116 in step 6304. The process D 6126 then identifies and authorizes the requester in step 6306. The process D 6126 checks if the requester is known in step 6308.

If the requester is known, the process D 6126 writes the request to process E service A signal tokenizer process 6122 in step 6310. The process E 6122 then replies with the state token and parameters to the process D 6126 in step 6312. The database system 522 then checks if a service session state object (SSO) 6128 exists for this session in step 6314. If the SSO 6128 does not exist, the database system 522 creates a SSO 6128 for this session and initializes the current state in step 6316. If the SSO 6118 exists, the database system 522 then proceeds to step 6318. In step 6218, the process D 6126 writes the state token and current state to process F—service state map process 6124 in step 6318. The process F 6124 then replies with a transition token and parameters to the process D 6126 in step 6320. The process D 6126 applies the transition token and the parameters to the SSO 6128 to update the current state and setting actions in step 6322.

In step 6324, the database system 522 checks if there is an action indicated by the SSO 6128. If there is not an action indicated by the SSO 6128, the database system 522 proceeds to step 6328. If there is an action indicated by the SSO

54

6128, the database system 522 performs the transition action specified by the SSO 6128 in step 6326. In step 6328, the database system 522 updates the session sliding usage accumulators. The database system 522 then checks if there is an action in the SSO 6128 for this usage in step 6330. If there is no action, the database system 522 returns to step 6302. If there is an action in the SSO 6128, the database system 522 performs the usage action specified in the SSO 6128 before returning to step 6302.

If the requester is not known, the database system 522 checks if there is a service A state object in step 6334. If there is a service state object, the database system 522 proceeds to step 6338. If there is no service state object, the database system 522 creates a service A state object and initializes the current state in step 6336. In step 6338, the database system 522 increments service A state object usage for this event. The database system 522 then checks if there is an action in the service A state object for this usage in step 6340. If there is no action in the service A state object for this usage, the database system 522 returns to step 6302. If there is an action in the service A state object for this usage, the database system 522 performs the usage action specified in the service A state object in step 6342 before returning to step 6342.

EXAMPLE

The following is one example of the access communication system that integrates many of the features described in the various embodiments of the inventions above. In the example below, a subscriber called Bank has many employees. The Bank has access to the network architecture 500 through an access provider called AccProv1. In this example, a bank employee is retrieving a stock quote from the yahoo.com website and then switches to request a streaming video from the yahoo.com website. Yahoo's web-server is running in the user network 562. Yahoo has access to the network architecture 500 through an access provider called AccProv2.

A system administrator for AccProv1 configures and optionally starts an access execution environment for the subscriber Bank in the local database system 570. The system administrator also starts a subscriber proxy for the Bank that runs in the access execution environment for the Bank. The subscriber proxy includes transport handlers, configuration handlers, and security handlers. The AccProv1 system administrator starts access execution environments for the maximum number of concurrent users in the local database system 570. The system administrator also starts generic proxies for users in the local database system 570. These generic proxies become user proxies after the user access profile is retrieved for the user.

The system administrator for AccProv2 starts an access execution environment for a service for Yahoo in the local database system 590. The system administrator also starts a service proxy for Yahoo's service that runs in the access execution environment for Yahoo. A system administrator for the Bank then sets up the user access profiles for each employee by inheriting attributes for the user access profile from capability classes or groups that the user belongs to. The user access profile includes the network shell for the user.

The Bank employee then logs on to the access server 524 using a user ID. The Bank system administrator set up. Besides user ID, a user may login to the access server 524 using a prepaid account code, a bank card number, an access card account code, or a user ID including a delimiter for user

access mobility. Upon connection, the access server 524 creates a device session in an access execution environment for the user. Then in response to the user login, the database system 522 retrieves the user access profile for the Bank employee. For a prepaid account code, a bank card number, an access card account code, and a user ID with a delimiter, the database system 522 may retrieve the user access profile from a database system external to the local database system 570. Also, the database system 522 may retrieve the user access profile from a database system external to the local database system 570 for global authentication.

Once the user access profile is retrieved, the access server 524 creates a user session in the access execution environment for the user and binds the device session with the user session. The configuration handler of the user proxy includes the network shell retrieved with the user access profile. The access server 524 then generates and transmits a list of available services in a service based directory to the Bank employee based on the user access profile.

The Bank employee then selects a service to check a stock quote for IBM by transmitting a request for service to the access server 524. The request for service may be in the form of a selection from the service based directory or an alias for alias translation for the network shell using domain name scoping or inbound DNS lookup. The access server 524 processes and transfers the request for service to the user proxy for the Bank employee. The user proxy transfers the request to the service proxy for Yahoo. The service proxy processes and transfers the request for service to Yahoo's web server. The database system 522 translates the service request for Yahoo to a private fulfillment address for one of Yahoo's web servers. The access server 554 then determines if a private destination address is available. If the private destination address is unavailable, the access server 554 could perform many actions including forwarding the request to another Yahoo web server or replying with a busy flag. The access server 524 also binds the user session and the device session with the selected service session by exchanging reference ID's.

The Bank employee then requests another stock quote for Oracle. The access server 524 then determines this stock quote was already cached based on the user's predictable pattern of requesting quotes for this company or a group of software companies. These prior requests for stock quotes of software companies by the Bank employee were audited to derive the Bank employee's predictable pattern. Alternatively, the request for the stock quote for Oracle could have been setup by the user in a script of commands to cache the quote in advance.

The Bank employee then requests a streaming video from Yahoo. The access server 524 switches the access for a premiere access based on the request for streaming video from the Bank employee. Alternatively, Yahoo initiates a switch of access for toll-free Internet service to the Bank employee who is a preferential customer. In order to be able to run the streaming video, the database system 522 also inherits user profile information from a class for streaming video user and updates the user access profile with the user profile information. The user proxy transfers the request for streaming video to the subscriber proxy. The subscriber proxy transfers the request to the service proxy. The service proxy then transfers the request to the Yahoo for one of their streaming video servers. Yahoo's streaming video server then provides the streaming video to the Bank employee.

Conclusion

The access network operates as described in response to instructions that are stored on storage media. The instruc-

tions are retrieved and executed by a processor in the access network. Typically, the processor resides in a server or database. Some examples of instructions are software, program code, and firmware. Some examples of storage media are memory devices, tape, disks, integrated circuits, and servers. The instructions are operational when executed by the processor to direct the network to operate in accord with the invention. The term "processor" refers to a single processing device or a group of inter-operational processing devices. Some examples of processors are computers, integrated circuits, and logic circuitry. Those skilled in the art are familiar with instructions, processors, and storage media.

Those skilled in the art will appreciate variations of the above-described embodiments that fall within the scope of the invention. As a result, the invention is not limited to the specific examples and illustrations discussed above, but only by the following claims and their equivalents.

1 claim:

1. A method of operating an access system including an access server to provide access between a user system and a plurality of communication networks that provide services to a user, the method comprising:

receiving a user login into the access server;

processing the user login to determine if the user is allowed access to the access system based on a local database system;

providing access to the access system to the user in response to the determination that the user is allowed access based on the local database system;

generating an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system;

in the second database system, receiving and processing the authorization query to determine whether the user is allowed access;

in the second database system, generating and transmitting an authorization response to the local database system;

receiving and processing the authorization response indicating whether the user is allowed to use the access system from the second database system; and

providing access to the access system to the user in response to the authorization response that allows the user to use the access system.

2. The method of claim 1 further comprising generating and transmitting a login query.

3. The method of claim 1 wherein processing the user login to determine if the user is allowed access to the access system based on a local database system is based on a user access profile.

4. The method of claim 1 further comprising determining if the second database system exists.

5. The method of claim 1 further comprising disconnecting a user's network device from the access server in response to the determination that the user is not allowed to use the access system.

6. The method of claim 1 wherein the login reply includes an access card account code.

7. The method of claim 1 wherein the login reply includes foreign network account information and further comprising identifying the second external database based on the foreign network account information.

8. The method of claim 1 further comprising:

receiving a request for access into a service control point;

processing the request for access to determine the destination for the request;

generating and transmitting a reply to route the request to the access server;

9. The method of claim 1 further comprising logging contract and settlement information.

10. An access system for providing access between a user system and a plurality of communication networks that provide services to a user, the access system comprising:

an access server connected to the user system and the plurality of communication networks and configured to receive and transmit a user logon from the user system to a local database system;

the local database system connected to the access server and configured to receive the user logon, process the user logon to determine if the user is allowed access to the access system based on the local database system, provide access to the access system to the user in response to the determination that the user is allowed access based on the local database system, generate an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system, receive and process an authorization response indicating whether the user is allowed to use the access system from the second database system, and provide access to the access system to the user in response to the authorization response that allows the user to use the access system; and

the second database system connected to the local database system and configured to receive and process the authorization query to determine whether the user is allowed access and generate and transmit the authorization response for the local database system.

11. The access system of claim 10 wherein the access server is configured to generate and transmit a logon query.

12. The access system of claim 10 wherein the local database system is configured to process the user logon to determine if the user is allowed access to the access system based on a user access profile.

13. The access system of claim 10 wherein the local database system is configured to determine if the second database system exists.

14. The access system of claim 10 wherein the access server is configured to disconnect a user's network device in response to the determination that the user is not allowed to use the access system.

15. The access system of claim 10 wherein the logon reply includes an access card account code.

16. The access system of claim 10 wherein the logon reply includes foreign network account information and the local database system is configured to identify the second external database based on the foreign network account information.

17. The access system of claim 10 further comprising:

a service control point connected to the user system configured to receive a request for access, process the request for access to determine the destination for the request, and generate and transmit a reply to route the request to the access server.

18. The access system of claim 10 wherein the local database system is configured to log contract and settlement information.

19. A software product for providing access between a user system and a plurality of communication networks that provide services to a user, the software product comprising:

database software operational when executed by a processor to direct the processor to receive the user logon, process the user logon to determine if the user is allowed access to an access system based on a local database system, provide access to the access system to the user in response to the determination that the user is allowed access based on the local database system, generate an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system, receive and process an authorization response indicating whether the user is allowed to use the access system from the second database system, and provide access to the access system to the user in response to the authorization response that allows the user to use the access system; and

a software storage medium operational to store the database software.

20. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to process the user logon to determine if the user is allowed access to the access system based on a user access profile.

21. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to determine if the second database system exists.

22. The software product of claim 19 wherein the logon reply includes an access card account code.

23. The software product of claim 19 wherein the logon reply includes foreign network account information and the database software is operational when executed by the processor to direct the processor to identify the second external database based on the foreign network account information.

24. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to log contract and settlement information.

* * * * *



US006784924B2

(12) **United States Patent**
Ward et al.

(10) **Patent No.:** US 6,784,924 B2
(45) **Date of Patent:** Aug. 31, 2004

(54) **NETWORK CONFIGURATION FILE FOR AUTOMATICALLY TRANSMITTING IMAGES FROM AN ELECTRONIC STILL CAMERA**

(75) Inventors: **Joseph Ward**, Hilton, NY (US);
Kenneth A. Parulski, Rochester, NY (US); **James D. Allen**, Rochester, NY (US)

(73) Assignee: **Eastman Kodak Company**, Rochester, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/004,046**

(22) Filed: **Jan. 7, 1998**

(65) **Prior Publication Data**

US 2003/0142215 A1 Jul. 31, 2003

Related U.S. Application Data

(60) Provisional application No. 60/037,963, filed on Feb. 20, 1997, and provisional application No. 60/037,962, filed on Feb. 20, 1997.

(51) Int. Cl.⁷ **H04N 5/225**

(52) U.S. CL. **348/207.1; 348/211.3**

(58) **Field of Search** 348/207, 211,
348/212, 213, 231, 232, 233, 552, 239,
211 99–211.13, 211.3, 17; 710/62, 3, 8;
345/327; 386/48, 117; 725/131–133, 139–141,
151–153, 709/217–219, 220, 705/26, 27,
455/566

(56) References Cited

U.S. PATENT DOCUMENTS

4,524,381 A 6/1985 Konishi
4,574,319 A 3/1986 Konishi
4,825,457 A 4/1989 Lebowitz
5,241,659 A 8/1993 Parulski et al.
5,477,264 A 12/1995 Sarbadhikari et al.

5,606,365 A 2/1997 Maurinus et al.
5,663,678 A 9/1997 Chang
5,666,159 A 9/1997 Parulski et al.
5,737,491 A * 4/1998 Allen et al. 704/270
5,800,005 A * 9/1998 Hull et al. 455/566
5,825,432 A * 10/1998 Yonezawa 348/563
6,111,604 A * 8/2000 Hashimoto et al. 348/220
6,122,005 A * 9/2000 Sasaki et al. 348/211.3
6,166,729 A * 12/2000 Acosta et al. 345/719
6,449,001 B1 * 9/2002 Levy et al. 348/1404

OTHER PUBLICATIONS

User's Manual—Axis 2420 Network Camera, 1996 (Axis Communications).*

U.S. patent application Ser. No. 60/037,963, Parulski et al. "The Universe of Smart Cards: EasySIM software", Schlumberger Limited.

"Claris Emailer" from Claris' home page on internet.

"Kodak Professional Digital Camera System—User's Manual," Eastman Kodak Company, 1991–1992, pp. x to xv, 2–1 to 2–3, 3–29 to 3–30, and 4–1 to 4–49.

* cited by examiner

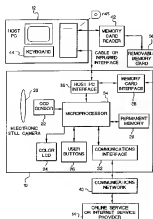
Primary Examiner—Tuan Ho

(74) Attorney, Agent, or Firm—Pamela R. Crocker

(57) ABSTRACT

A network configuration file is generated at a host computer and downloaded to a digital camera. This file contains instruction information for communicating with a selected destination via a communications interface. The digital camera includes a "send" button or LCD icon which allows the user to easily transmit one or more images via a wired or wireless communications interface to a desired destination, which among other possibilities may be an Internet Service Provider or a digital photofinishing center. When the user selects this option, the communications port settings, user account specifics, and destination connection commands are read from the network configuration file on the removable memory card. Examples of these settings include serial port baud rate, parity, and stop bits, as well as account name and password.

9 Claims, 4 Drawing Sheets



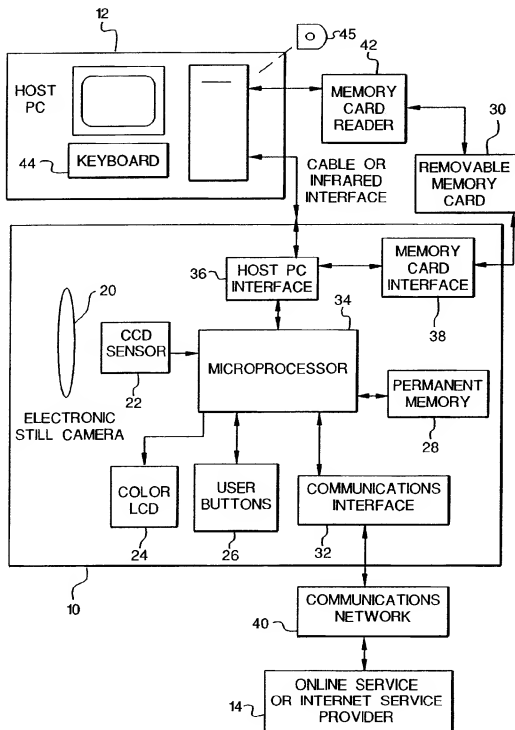
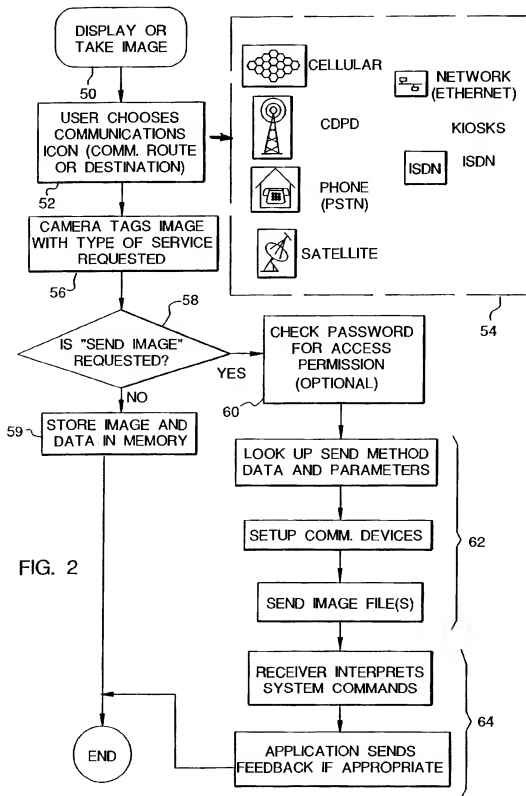


FIG. 1



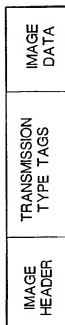


FIG. 3



FIG. 4

FIG. 4A

PHONE (PUBLIC SWITCHED TELEPHONE NETWORK)



PROTOCOL TYPE	PHONE NUMBER	DEFAULT SETTINGS	MODERN CONTROL STRING	ACCOUNT DATA	PASSWORD	SYSTEM COMMANDS
------------------	-----------------	---------------------	-----------------------------	-----------------	----------	--------------------

CELLULAR OR PCS

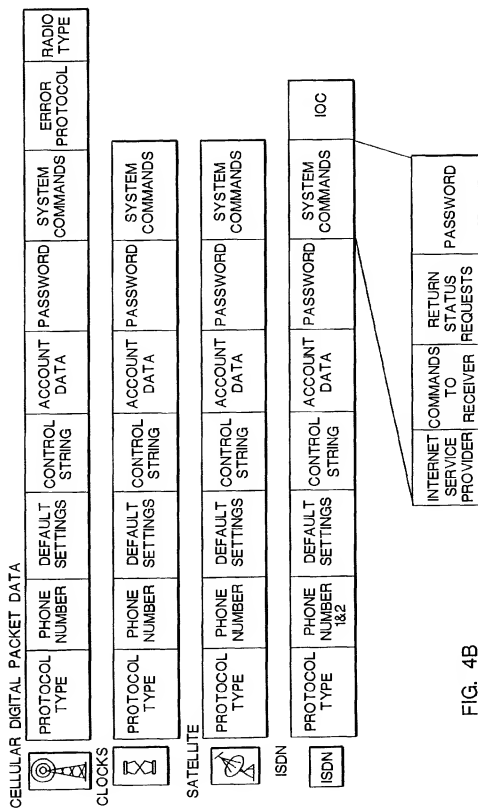


PROTOCOL TYPE	PHONE NUMBER	DEFAULT SETTINGS	MODERN CONTROL STRING	ACCOUNT DATA	PASSWORD	SYSTEM COMMANDS	ERROR PROTOCOL	RADIO TYPE
------------------	-----------------	---------------------	-----------------------------	-----------------	----------	--------------------	-------------------	---------------

WIRELESS LAN



PROTOCOL TYPE	PHONE NUMBER	DEFAULT SETTINGS	PARA- METER FILE	ACCOUNT DATA	PASSWORD	SYSTEM COMMANDS
------------------	-----------------	---------------------	------------------------	-----------------	----------	--------------------



1

NETWORK CONFIGURATION FILE FOR AUTOMATICALLY TRANSMITTING IMAGES FROM AN ELECTRONIC STILL CAMERA

CROSS-REFERENCE TO RELATED APPLICATION(S)

THIS APPLICATION CLAIMS BENEFIT OF PROVI-
SIONAL APPLICATION SERIAL NO. 60/037,962 FILED
Feb. 20, 1997.

Reference is made to commonly assigned copending
applications Serial No. 60/037,963, filed Feb. 20, 1997
entitled "Electronic Camera with 'Utilization' Selection
Capability" and filed in the names of Kenneth A. Parulski,
Joseph Ward, and Michael C. Hopwood, which is assigned
to the assignee of this application.

FIELD OF THE INVENTION

The invention relates generally to the field of
photography, and in particular to electronic photography.
More specifically, the invention relates to a digital camera
that interfaces with a host computer.

BACKGROUND OF THE INVENTION

Digital cameras, such as the Kodak Digital Science
DC25™ camera, allow images to be utilized on a home
computer (PC) and to be incorporated into e-mail documents
and personal home pages on the World Wide Web. Presently,
images must be copied to the PC and transmitted as e-mail,
for example using an online service or an Internet Service
Provider (ISP), via a modem from the user's PC. It would be
desirable to be able to transmit pictures directly from the
digital camera instead of first transferring the pictures to a
PC. For instance, on a vacation trip, it is desirable to
immediately share pictures with friends or relatives via
e-mail or Internet access. It is also desirable to transmit
pictures from a location without PC access in order to free
up camera storage to take additional pictures. There are a
wide variety of connection means to online services such as
America On Line, ISPs, and bulletin board services. Each of
these services typically requires an account name and
password, as well as local telephone access numbers, and
specific communications settings. It would be difficult to
provide an easy-to-use means with buttons or menus on a
small digital camera to input and/or modify all of these
required settings.

SUMMARY OF THE INVENTION

The present invention is directed to overcoming one or
more of the problems set forth above. Briefly summarized,
according to one aspect of the present invention, a network
configuration file is generated at a host computer and
downloaded to a digital camera. This file contains instruction
information for communicating with a selected destina-
tion via a communications interface. The digital camera
includes a "send" button or LCD icon which allows the user
to easily transmit one or more images via a wired or wireless
communications interface to a desired destination, which
among other possibilities may be an Internet Service Pro-
vider or a digital photofinishing center. When the user
selects this option, the communications port settings, user
account specifics, and destination connection commands are
read from the network configuration file. Examples of these
settings include serial port baud rate, parity, and stop bits, as
well as account name and password.

2

In addition, information about which image or images to
transmit is entered using the user buttons on the digital
camera. This information is used to automatically establish
a connection, log-in to the desired destination, and to
transmit the image. The transmission may occur immedi-
ately after the pictures are taken, for example if the camera
has a built-in cellular phone modem, or at a later time, when
the camera is connected to a separate unit (such as a dock,
kiosk, PC, etc.) equipped with a modem. In the latter case,
a "utilization file" is created to provide information on
which images should be transmitted to which account.

These and other aspects, objects, features and advantages
of the present invention will be more clearly understood and
appreciated from a review of the following detailed descrip-
tion of the preferred embodiments and appended claims, and
by reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a system block diagram of the invention.
FIG. 2 is a diagram showing the steps used to automati-
cally transmit images using the network configuration file.
FIG. 3 is a diagram of an image file.
FIG. 4 is a diagram showing several versions of the
network configuration file.

DETAILED DESCRIPTION OF THE INVENTION

Because imaging systems and devices are well known, the
present description will be directed in particular to elements
forming part of, or cooperating more directly with, apparatus
in accordance with the present invention. Elements not
specifically shown or described herein may be selected from
those known in the art. Some aspects of the present descrip-
tion may be implemented in software. Unless otherwise
specified, all software implementation is conventional and
within the ordinary skill in the programming arts.

A system block diagram of the invention is shown in FIG.
1 including an electronic still camera 10, a host computer
(PC) 12 and a service provider 14. The camera includes an
optical section 20 for imaging a scene upon a CCD sensor
22 and generating an image signal, a liquid crystal display
(LCD) 24 for displaying images and other information, a
number of user input buttons 26, both permanent memory 28
and removable memory 30, and an internal communications
interface 32 (e.g., modem). This interface may connect to a
variety of known networks, such as a public switched
telephone network (PSTN), ISDN, an RF cellular phone
network, or Ethernet. The camera 10 also includes a micro-
processor 34 for generally controlling the camera functions,
as well as the interchange of data with the host PC 12 and
the memory card 30 through a host PC interface 36 and a
memory card interface 38, respectively. Besides the host PC
12, the system includes a network connection 40 to the
online service or ISP (Internet Service Provider) 14.
Alternatively, the network 40 can connect to the user's home
PC 12.

When the camera 10 is first purchased (or at any time
thereafter), it is connected to the PC 12 via the host PC 36
interface and a software application (stored on a disc 45)
running on the host PC 12 will enable the user to specify the
name of a destination ISP or online service and to input from
the host PC keyboard 44 the appropriate communication
settings and account information. This information gener-
ates a network configuration file, which then can be then be
downloaded to the camera 10 through the host PC interface

3

4

36, which may be a wired or infrared (e.g., IrDA) interface, and written to the camera's internal memory 28 and/or the removable memory card 30. Alternatively, a host PC equipped with a memory card reader/writer 42 can write the information directly to the card 30 without connecting the camera through its host PC interface 36. Also, this information could be predetermined by the user and stored in a "preferences" file on the host PC 12 and then transferred to the camera 10 from this file without further intervention by the user. Multiple sets of destination services can be stored on the memory card 30. Typically, keyword or graphic descriptors (e.g., icons) accompany the information in the network configuration file about destination services to enable easy access by the camera user.

The steps used to automatically transmit images using the network configuration file are shown in FIG. 2. After disconnecting the camera from the host PC, the user operates the camera to take pictures (step 50). This is typically done at a remote location, for example while traveling to another city. As the user takes or reviews images on the image LCD display, the decision can be made to transmit one or more images (step 52). This is done by choosing one of the keywords or icons in a menu 54 shown in FIG. 2, which are displayed on the LCD 24 and selected, e.g., through the user buttons 26. (Note that a camera will typically only include a subset (only those desired by the user) of all the different services shown.) The selected image files may be tagged with a code (step 56) indicating which service is requested, as shown in FIG. 3. (Alternately, an "image utilization" file can be created in the camera storing a list of images to be transmitted by a particular method, as described in the cross-referenced depending patent application (U.S. Serial No. 60/037,963). As described in that patent application, the details of an order, e.g., number of print copies to be made from an image and the size of the prints and/or a list of images to be e-mailed to various recipients, is written into the "utilization" file, which identifies the order and includes pointers to the image files that store the images required to "fulfill" the order. The "utilization" file is stored in the internal memory 28 or the memory card 30.)

Next, the system determines whether a request exists to send an image (step 58). If no request is present, the image and associated data is stored in either permanent memory 28 or the memory card 30 (step 59). (Typically, all images are initially saved in memory whether eventually sent or not.) Otherwise, if there is a request to send an image, the user ensures that the camera is connected to the appropriate service (wired telephone line, cellular phone, kiosk, etc.) and pushes a "send" button in the user button section 26, or selects a "send" menu option on the LCD 24. The camera then utilizes the appropriate network configuration file, shown in FIG. 4. Each network configuration file contains items such as the protocol type, phone number, etc., as described in Appendix I. The user password may be checked against the password in the network configuration file to ensure that the user is authorized to connect the camera to the desired service (step 60). Alternately, the stored password in the appropriate configuration file can be used. Next, the camera uses the parameters in the configuration file to establish communications with the service and send one or more image files as selected by the user (steps 62). The service receiver interprets the system commands issued by the camera from the network configuration file list and sends appropriate feedback (such as "transfer in progress" and "transfer complete") which are interpreted by the camera and displayed on the LCD 24 (steps 64).

For example, when the camera uses a normal wired telephone (Public Switched Telephone Network) connection

(i.e., network 40) to the camera's internal modem 32, after the user selects the images to be sent and presses the "send" button, the camera performs the following steps without user intervention:

- 1) Read the appropriate connection parameters from the network configuration file (on the memory card 30 or internal camera memory 28), dial the phone and establish the connection to the destination service 14.
- 2) Read the user's account name and password and transmit these to "log-on" to the service 14.
- 3) Using the appropriate communications protocol (FTP, mailto, etc.), transmit the selected image or images to the destination service 14.

The invention has been described with reference to a preferred embodiment. However, it will be appreciated that variations and modifications can be effected by a person of ordinary skill in the art without departing from the scope of the invention.

Appendix I

These are descriptions of the tags listed in the previous drawing:

Protocol Type

Each communication method has its own protocol, or rules to communicate. This tag identifies that protocol and where to find it. For example, the Network may use TCP/IP and a modem may use XModem.

Phone Number

This is the number of the receiving service. If internet access is requested, this could be the number of the Internet Service Provider. For ISDN, some systems require two phone numbers, dialed and connected to in sequence.

Default Settings

Standard settings that make the communications device compatible with the imaging device.

Modem Control String

Modem and communications devices have a command language that can set them up before they are used. For example, modems have many options controlled by command strings including volume level, the amount of time the carrier is allowed to fail before the system hangs up, and so on.

Account Data

This can be internet account data, charge number data, phone card data, billing address, and data related to the commerce part of the transmission.

Password

Any password needed to get into the communications system. Other passwords to get into the remote application or destination are located in the System Commands section.

System Commands

These are commands that control the end destination.

Error Protocol

In cellular and some other wireless communications, error protocols are used to increase the robustness of the link. For example, MNP10 or ETC may be used for cellular links.

Radio Type

The type of radio used for this communications feature may be identified here. Some cell phones have modems built in, others will have protocols for many communications functions built in. The radio type will make the imaging device adapt to the correct interface.

IOC

ISDN Ordering Code identifies what features are available on the ISDN line provided by the telco. It is used to establish the feature set for that communications link.

5

Internet Service Provider

This identifies the actual service provider and any specific information or sequence of information that the service wants to see during connection and logoff. It also tells the device how to handle the return messages, like "time used" 5 that are returned by the server.

Commands to Receiver

This may be a list of commands to control the receiving application. For example, a command to print one of the images and save the data to a particular file on a PC may be embedded here. 10

Return Status Requests

This tag can set up the ability of the application to tell if an error has occurred, or what the status of the application might be. The data here will help the device decide if it should continue communicating and a set user interface response can be developed around this feedback. 15

What is claimed is:

1. A method for directing the automatic transmission of images from an electronic camera, said method including the steps of: 20

storing network configuration information in a memory of the electronic camera;

using the network configuration information to connect the camera to an online service or internet service provider (ISP); 25

transferring pictures and data from the camera to the online service or ISP;

receiving information from the online service or ISP indicating that the transfer of the pictures and data is complete; and 30

displaying information on a display of the electronic camera indicating that the transfer of the pictures and data is complete; 35

wherein the network configuration information is generated at least in part in a host device adaptable for communication with the electronic camera, based on information previously stored in the host device and utilizable to connect the host device to the online service or ISP. 40

2. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes a PSTN (Public Switched Telephone Network).

3. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes an ISDN network. 45

4. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes an RF cellular phone network.

6

5. The method as claimed in claim 1 wherein the connection between the camera and the online service or ISP includes an Ethernet connection.

6. The method of claim 1 wherein the information previously stored in the host device comprises information stored in a network configuration preference file of the host device.

7. A method for directing the automatic transmission of images from an electronic camera, said method including the steps of:

storing network configuration information in a memory of the electronic camera, the network configuration information comprising information for using a communications network to establish a connection between the electronic camera and an online service or internet service provider (ISP); and

in response to user activation in the electronic camera of a transmission command specifying one or more pictures for transmission, automatically performing without further user intervention the steps of:

retrieving the stored network configuration information;

utilizing the retrieved information to establish a connection between the electronic camera and the online service or ISP; 25

transferring the one or more pictures from the electronic camera to the online service or ISP;

receiving information from the online service or ISP indicating that the transfer of the one or more pictures is complete; and

displaying the information on a display of the electronic camera indicating that the transfer of the one or more pictures is complete; 30

wherein the network configuration information is generated at least in part in a host device based on information previously stored in the host device, and the generated network configuration information is subsequently downloaded from the host device into the electronic camera in conjunction with the storing step. 35

8. The method of claim 7 wherein the host device comprises a personal computer.

9. The method of claim 7 wherein the information previously stored in the host device comprises information stored in a network configuration preference file of the host device. 40

* * * * *



US006636259B1

(12) **United States Patent**
Anderson et al.

(10) **Patent No.:** **US 6,636,259 B1**
(45) Date of Patent: **Oct. 21, 2003**

- (54) **AUTOMATICALLY CONFIGURING A WEB-ENABLED DIGITAL CAMERA TO ACCESS THE INTERNET**
- (75) Inventors: **Eric C. Anderson**, San Jose, CA (US);
Robert Paul Morris, Raleigh, NC (US)
- (73) Assignee: **IPAC Acquisition Subsidiary I, LLC**,
Peterborough, NH (US)
- (*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 32 days.
- (21) Appl. No.: **09/625,824**
- (22) Filed: **Jul. 26, 2000**
- (51) Int. Cl.⁷ **H04N 5/232**
- (52) U.S. Cl. **348/211.3; 348/211.12; 348/14.04**
- (58) **Field of Search** **348/14.01, 14.02, 348/14.04, 14.08, 14.09, 552, 143, 156, 157, 211.3, 211.12; 709/322**
- (56) **References Cited**

U.S. PATENT DOCUMENTS

- 5,430,827 A * 7/1995 Rissanen 704/272
5,737,491 A * 4/1998 Allen et al. 704/270
5,905,736 A * 5/1999 Ronen et al. 370/546
6,064,671 A * 5/2000 Killian 370/389
6,067,571 A * 5/2000 Igarashi et al. 709/232

- 6,167,469 A * 12/2000 Safai et al. 710/62
6,226,752 B1 * 5/2001 Gupta et al. 713/201
6,269,481 B1 * 7/2001 Perlman et al. 717/11
6,502,195 B1 * 12/2002 Colvin 713/202
- * cited by examiner

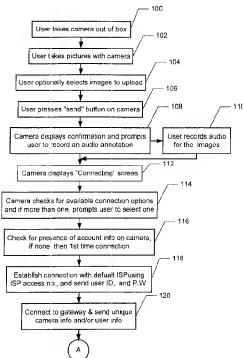
Primary Examiner—Wendy R. Garber
Assistant Examiner—Jacqueline Wilson

(74) *Attorney, Agent, or Firm*—Sawyer Law Group LLP

(57) ABSTRACT

The present invention provides a method and system for automatically establishing a user account and for configuring a hand-held electronic device for accessing a site on a public network using the user account. The method and system include establishing a connection to a website server, which is responsible for establishing and maintaining website accounts. The presence of account information on the electronic device camera is then checked, and if none are found, it is determined that this is a first-time connection and information uniquely identifying the electronic device is sent to the website server so that the server can set-up the account information based on the electronic device information. The server then sends user account information to the device, including an account ID and password. The user account information is then stored on electronic device for use the next time the electronic device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network.

22 Claims, 6 Drawing Sheets



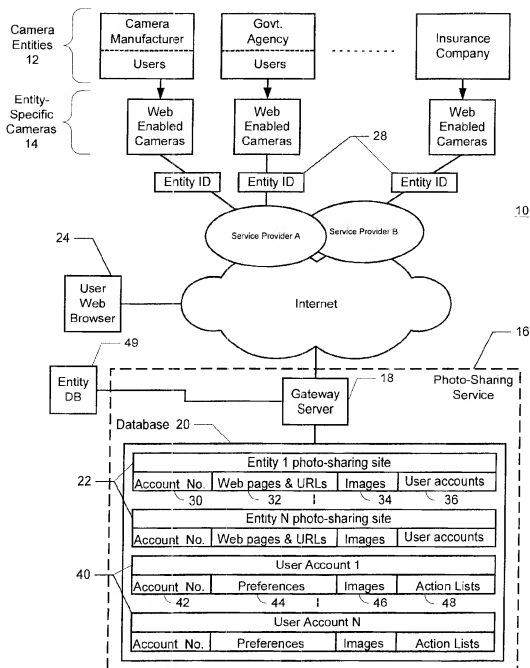


FIG. 1

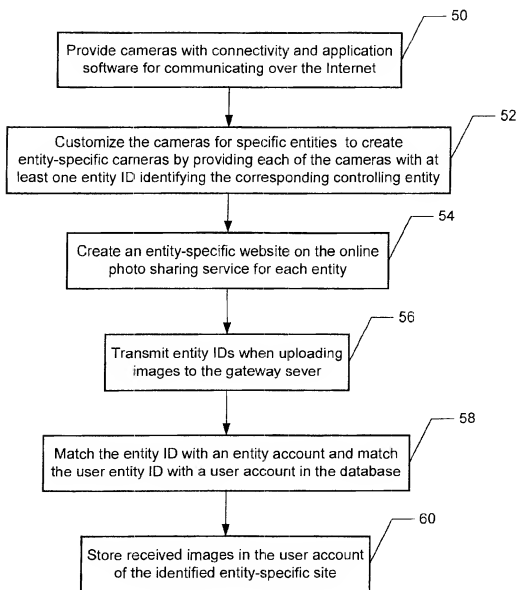


FIG. 2

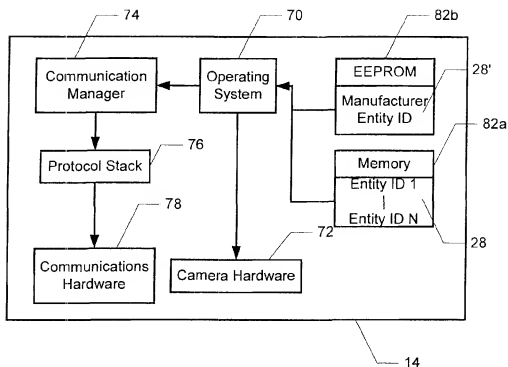


FIG. 3

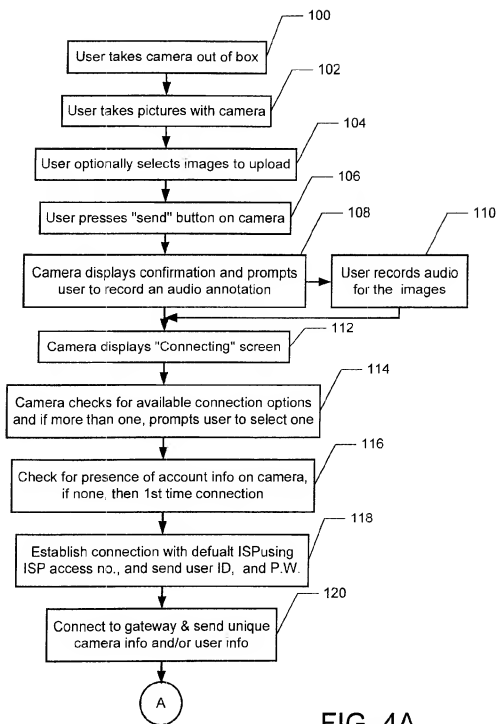


FIG. 4A

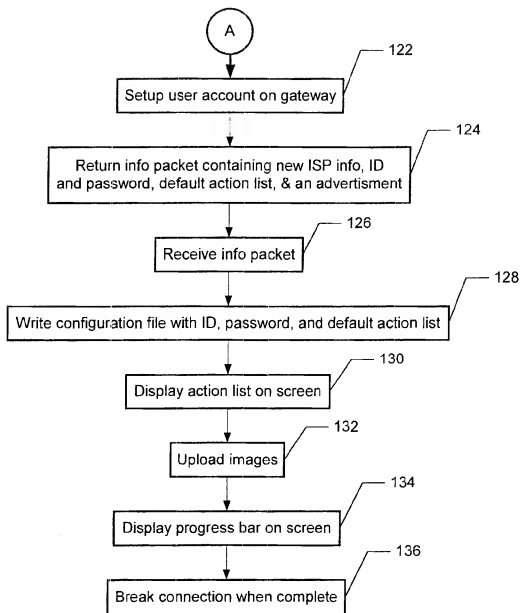


FIG. 4B

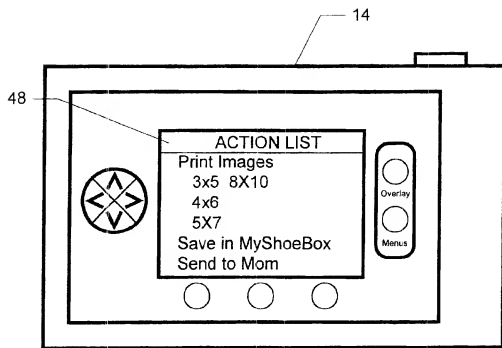


FIG. 5

1

AUTOMATICALLY CONFIGURING A WEB-ENABLED DIGITAL CAMERA TO ACCESS THE INTERNET

CROSS-REFERENCE TO RELATED APPLICATIONS

The present invention is related to co-pending U.S. patent application Ser. No. 09/625,398 entitled "Method and System For Hosting Entity-Specific Photo-Sharing Websites For Entity-Specific Digital Cameras"; and to co-pending U.S. patent application Ser. No. 09/626,418, entitled "Method And System For Selecting Actions To Be Taken By A Server When Uploading Images," which are assigned to the assignee of the present application and filed on the same date as the present application.

FIELD OF THE INVENTION

The present invention relates to a method and system for customizing digital cameras to upload images to an entity-specific photo-sharing websites, and more particularly to automatically configuring a web-enabled digital camera to access the Internet.

BACKGROUND

As the popularity of digital cameras grows, the desire of digital camera users to share their images with others will also continue to grow. New digital camera owners typically try to share their images based on the paradigm of film cameras, in which images are printed on paper and then placed into a photo album. The most straightforward approach to do this with a digital camera is to connect the digital camera directly to a printer to create the prints, and then manually insert the images into a photo album. Users often find this process somewhat complicated and restrictive because standard printers can only print images in limited sizes and requires particular types of paper. And even after the photo album has been assembled, the printed images are not easily shared with many people.

The best approaches to photo-sharing take advantage of the Internet. One such approach is for users to store the digital images on a PC and then send the images to others using email. Several Internet companies now offer an even more convenient approach by providing photo-sharing websites that allow users to store their images for free and to arrange the images into web-based photo albums. Once posted on a photo-sharing website, others may view the images over the Internet.

While convenient for storing digital images, getting the images to the photo-sharing websites can be challenging for users. Most commonly, users must upload their images from the digital camera to a PC using a cable or IrDA, or by inserting the camera's flash card into the PC. From the PC, the user logs onto the Internet and uploads the images to a photo-sharing website. After uploading the images, the user works on the website to arrange the images into web albums and to add any textual information.

Although today's approach for storing images from a digital camera onto a web photo-sharing website and for creating web photo albums works reasonably well, two problems remain that hinder the mainstream adoption of web-based photo-sharing. One problem is that this approach requires the use of a PC, notebook computer, or PDA. While many digital camera users today have PC's, most of those users are early adopters of technology. There are many other consumers who would purchase a digital camera, but are

2

reluctant to do so because they do not yet own a PC or are intimidated by them.

In an effort to address this problem, the assignee of the present application developed an approach to uploading images to the web that doesn't require the use of a PC. In this approach, an email software application is loaded into a digital camera capable of running software that allows the user to e-mail the images directly from the camera. The user simply connects his or her digital camera to a cellphone or modem, runs the e-mail application, and selects the desired images and the email recipients. The selected images are then sent to the recipients as e-mail attachments.

Although emailing photos directly from the camera allows users who do not own a PC to share images over the Internet, these users must still establish accounts with both an Internet service provider (ISP) and the photo-sharing website before being able to post their images. These accounts must also be set-up by PC users as well. For techno savvy users who use a PC to upload the images to the photo-sharing website, establishing the accounts may not be a bother, but even these users may not always have their PC's handy, such as when on vacation, for instance. And for non-PC users, establishing the accounts by entering account information on the digital camera itself may prove to be a cumbersome, if not a daunting, task.

Accordingly, what is needed is an improved method for uploading images from a digital camera to a photo-sharing website on the Internet. In order for online photo-sharing to become more mainstream, an approach that doesn't require a PC or PC expertise and that reduces complexity for the user is required. The present invention addresses such a need.

SUMMARY OF THE INVENTION

The present invention provides a method and system for automatically establishing a user account and for configuring a hand-held electronic device for accessing a site on a public network using the user account. The method and system include establishing a connection to a website server, which is responsible for establishing and maintaining website accounts. The presence of account information on the electronic device camera is then checked, and if none are found, it is determined that this is a first-time connection and information uniquely identifying the electronic device is sent to the website server so that the server can set-up the account information based on the electronic device information. The server then sends user account information to the device, including an account ID and password. The user account information is then stored on electronic device for use the next time the electronic device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network.

According to the method and system disclosed herein, a user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network with the electronic device.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG 1 is a block diagram illustrating an online photo-sharing system in accordance with a preferred embodiment of the present invention.

FIG 2 is a flow chart illustrating a business method for hosting entity-specific photo-sharing websites for entity-specific image capture devices in accordance with a preferred embodiment of the present invention.

3

FIG. 3 is a diagram showing a preferred embodiment of the connectivity and application software of the camera.

FIGS. 4A and 4B illustrate a flow chart of a process for automatically configuring a web-enabled digital camera to access the Internet in a preferred embodiment of the present invention.

FIG. 5 is a block diagram illustrating an example action list 48 displayed on the LCD screen of the camera.

DETAILED DESCRIPTION

The present invention relates to an automatic system for uploading images from a digital camera to entity-specific photo-sharing websites and for automatically establishing accounts. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

FIG. 1 is a block diagram illustrating an online photo-sharing system 10 in accordance with a preferred embodiment of the present invention. The system includes a plurality of camera controlling entities 12 that produce, own, or otherwise control a set of digital cameras 14, and an online photo-sharing service 16. The online photo-sharing service 16 includes a gateway server 18 and an entity/account database 20. The various camera controlling entities 12 contract with the photo-sharing service 16 to transparently host customized photo-sharing websites 22 for each entity, which are referred to herein as entity-specific photo-sharing websites 22. The entity-specific photo-sharing websites 22 are each accessible to the user through the entity's existing Internet site (not shown), and thus appear to users as though the entity-specific photo-sharing websites 22 are hosted by the corresponding entity. According to a preferred embodiment of the present invention, the cameras 14 for a particular entity are customized for that entity to create entity-specific cameras 14, such that when the cameras 14 connect to the Internet, the cameras 14 automatically upload their images to the photo-sharing website of the corresponding entity. In a further aspect of the present invention, the photo-sharing service 16 automatically stores the images in a web album, which is viewable over the Internet by a user's web browser 24.

As used herein, a camera controlling entity 12 is any entity that makes, owns, sells, or controls digital cameras 14, and therefore includes, camera manufacturers, companies, retailers, and end-users. One or more combination of these entities 12 may either contract with the photo-sharing service 16 to provide entity-specific websites 22 for their cameras 14, or have entity information transmitted to the photo-sharing service 16 from the cameras 14. Therefore, a camera controlling entity 12 may include a single entity 12 or a hierarchical relationship of entities 12.

An example of a single entity 12 includes an insurance company that contracts with the photo-sharing service 16 to have all digital cameras 14 used by their agents to transmit their images to a customized insurance photo-sharing website. Examples of a hierarchical relationships of entities 12 includes a camera manufacturer, such as Nikon, that contracts with the photo-sharing service 16 to have all Nikon digital cameras 14 transmit their images to the customized

4

Nikon photo-sharing website. Since the images of different users must be distinguished, each user of a Nikon camera 14 would also constitute an entity within the Nikon website so that the images from different users can be distinguished. Other examples of hierarchical entity relationships include a retailer and its consumers, a real estate agency and its agents, community groups and its members, and government agencies and its employees, for instance.

In a preferred embodiment, the cameras 14 are customized for their respective entities 12 by providing the cameras 14 with software for transmitting entity ID information 28 identifying its controlling entity 12 to the photo-sharing service 16. The photo-sharing service 16 in conjunction with the gateway server 18 and the entity/account database 20 hosts the entity-specific photo-sharing websites 22. Each entity-specific website 22 is identified in the database 20 by an entity account number 30 and includes the web pages and URIs 32 comprising the website, the images and web albums 34 stored on the website, and the user account numbers 36 of authorized users. The database 20 also includes user accounts 40, each of which comprises a user account number 42, user preferences 44, the user's images 46, and action lists 48, explained further below.

The gateway server 18, which communicates with the cameras 14 during image uploading, receives one or more entity IDs 28 from each camera 14 and matches the entity ID 28 with an entity account 30 in the database 20. The images are then automatically associated with the photo-sharing website 22 of the identified entity 12 and/or the identified user.

After the images are uploaded, a user of the camera 14 may visit the online photo-sharing website 22 over the Internet to view the images via a web browser 24. Since the photo-sharing websites 22 are transparently hosted by the photo-sharing service 16, each photo-sharing website 22 appears as though it is hosted by the entity itself, rather than the third party service.

In one embodiment, the cameras 14 may connect to the Internet via a service provider 26, which may include a wireless carrier and/or an Internet service provider (ISP) that is capable of servicing many devices simultaneously. In a preferred embodiment, each of the cameras 14 is provided with wireless connectivity for connecting to the Internet, and are therefore so called "web-enabled" devices, although a wired connection method may also be used.

The cameras 14 may be provided with wireless connectivity using anyone of a variety of methods. For example, a cellphone may be used to provide the digital camera 14 with wireless capability, where the camera 14 is connected to the cellphone via a cable or some short-range wireless communication, such as Bluetooth. Alternatively, the camera 14 could be provided with built-in cellphone-like wireless communication. In an alternative embodiment, the digital camera 14 is not wireless, but instead uses a modem for Internet connectivity. The modem could be external or internal. If external, the camera 14 could be coupled to modem via any of several communications means (e.g., USB, IEEE1394, infrared link, etc.). An internal modem could be implemented directly within the electronics of camera 14 (e.g., via a modem ASIC), or alternatively, as a software-only modem executing on a processor within camera. As such, it should be appreciated that, at the hardware connectivity level, the Internet connection can take several forms. Hence, it should be appreciated that the present invention is not limited to any particular method of accessing the Internet.

5

In a preferred embodiment, the entity-specific websites 22 are customized to seamlessly integrate into the entity's existing website by following the look and feel of the entity's existing website. The entity-specific websites 22 are hosted on the photo-sharing service 16, but a link to the entity-specific websites 22 may be provided on the homepage of the corresponding entity's existing website. Thus, in order to view a web album on an entity-specific website 22, the user must visit the entity's existing website and click the link to the entity-specific website 22, where the user's browser 24 will be transparently directed to the photo-sharing service 16 and be provided with the web pages 32 of the entity-specific website 22.

As an example of the operation of the photo sharing system 10, consider the following scenario. Assume that Minolta and Nikon are entities 12 that have contracted with the photo-sharing service 16, and that the photo-sharing service 16 hosts a photo-sharing website 22 for Minolta and a photo-sharing website 22 Nikon. The Minolta cameras 14 would be provided the entity ID 28 for Minolta and the Nikon cameras 14 would be provided the entity ID 28 for Nikon. When the Minolta and the Nikon cameras 14 send sets of images to the photo-sharing service 16, the gateway server 18 would distinguish the cameras 14 by the entity IDs 28 and would direct the set of images received from Minolta cameras 14 to Minolta's photo-sharing website, and would direct the images from Nikon cameras 14 to Nikon's photo-sharing website. To view the images, the owners of the cameras 14 would use a browser 24 on their PC or PDA to visit the URL of the Minolta or Nikon photo-sharing websites 22. In one preferred embodiment, the photo-sharing service 16 sends the URL of the entity-specific website 22 directly to the camera 14 for display to inform the user of the address.

According to the present invention, the photo-sharing service 16 provides business-to-business and business-to-consumer business models. The service is business-to-business because the service provides companies, such as camera manufacturers, with a complete end-to-end solution for their cameras 14. The solution includes customized software for their cameras 14 for sending images over the internet, and an internet website for storing images from those cameras 14 on a branded website that appears to be hosted by the company. The service is business-to-consumer because it allows users of digital cameras 14 with an automatic solution for uploading captured images from a digital camera 14 to an online photo-sharing website, without a PC.

According to one preferred embodiment of the present invention, the photo-sharing service 16 provides a method of doing business whereby the photo-sharing service 16 shares revenue based on the hierarchical relationship of the entities 12. For example, if the photo sharing service 16 charges a fee for receiving and/or storing the images received from the entity-specific cameras 14, then the photo sharing service 16 may share a portion of the fee with the manufacturer and/or third party supplier of the camera 14 that uploaded the images, for instance. Revenue may also be shared with the wireless service provider providing the connection with the photo sharing service 16.

FIG. 2 is a flow chart illustrating a business method for hosting entity-specific photo-sharing websites for entity-specific image capture devices, such as digital cameras, in accordance with a preferred embodiment of the present invention. First, the cameras 14 are provided with connectivity and application software for communicating over the Internet in step 50. In a preferred embodiment, this step is

6

performed during camera 14 manufacturing to provide off-the-shelf web enabled cameras 14. The cameras 14 are also customized for specific entities 12 to create entity-specific cameras 14 by providing each of the cameras 14 with at least one entity ID 28 identifying the corresponding controlling entity in step 52.

Referring now to FIG. 3, a diagram showing the preferred embodiment of the connectivity and application software of the camera 14 and the entity ID 28 information is shown. Preferably, the camera 14 includes a microprocessor-based architecture that runs an operating system 70 for controlling camera hardware 72 and overall functionality of the camera 14 (e.g., taking pictures, storing pictures, and the like). An example of such an operating system 70 is the Digita™ Operating Environment developed by the assignee of the present application. The camera 14 also includes communication manager 74 software, and a TCP/IP protocol stack 76, that enables communication via the internet, as is well-known in the art. The entity ID information 28 and captured images may be stored in one or more types of memories 82.

For hierarchical entity relationships, the cameras 14 are provided with hierarchical entity IDs 28; one entity ID 28 identifying the entity, and a second entity ID 28 identifying the end-user. Whether there are one or more entity IDs 28, the entity ID 28 of the camera manufacturer may always be provided. Camera 14 customization may occur either during manufacture or anytime thereafter. In a preferred embodiment, the manufacturer entity ID 28 is provided at the time of manufacturing and is stored in an EEPROM 82b, while the entity IDs 28 for other entities 12, such as companies and end-users, are loaded into the camera 14 subsequent to manufacturing and are stored in flash memory 82a or the EEPROM 82b.

Customization that occurs subsequent to manufacture may be implemented using several methods. The first method is to manufacture the cameras 14 with an application programming interface (not shown) for accepting a subsequently loaded software application that specifies the entity ID's 28. The application may come preloaded on a flash card, which is then inserted into the camera 14 by the user and stored in flash memory 82a. The application may also be wirelessly beamed into the camera. When executed in the camera, the software application transmits the appropriate entity IDs 28 to the gateway server 18.

The second method is to load a small file in the camera 14 specifying the entity IDs 28 from a removable memory or from a PC, and storing the file in a system folder within the camera's flash memory 82a. The camera 14 then accesses the file when an Internet connection is established. In a preferred embodiment, the communication manager 74 automatically extracts the manufacturing ID 28 and the entity ID 28 and transmits them to the gateway server 18. In this embodiment, the entity ID 28 is also stored in the EEPROM 82b and is factory set to zero (empty). Thus, unless the entity ID 28 is set, the manufacturing ID 28 may default as the highest controlling entity.

If, for example, a third party developer X contracts to provide custom camera software for camera manufacturer Z, then a custom entity ID will be issued for developer X and developer X will place the custom entity ID into the EEPROM 82b. Developer X is now a controlling entity 12, and may specify to the photo-sharing service 16 that a developer X entity-specific photo-sharing site 22 or developer X's own website be the destination for the uploaded images.

7

The protocol stack 76, under direction of the communications manager interfaces with the communications hardware 78 of camera. The protocol stack 76 includes software APIs and protocol libraries that interface with the communication manager 74, and communication hardware interface drivers that interfaces directly with the various communication hardware 72 the camera 14 must function with (e.g., a Bluetooth transceiver, etc.). The communication manager 74 communicates with operating system 70 and the IP protocol stack 76 to establish an Internet connection ant to transmit the entity ID 28 information and images from the memories 82a and 82b to the photo-sharing service 16.

In an alternative embodiment, rather than loading entity ID's 28 into the camera, a combination of the camera's serial number and the make and model number of the camera may be used as the entity ID 28. Entity specific cameras 14 may then be distinguished by providing a mapping of the camera serial numbers and product IDs to specific entities 12 in the database 20.

Although the camera 14 has been described in terms of a software-based customization solution, those with ordinary skill in the art will readily recognize that the camera 14 may also be provided with a hardware-based solution.

Referring again to FIG. 2, before or after camera customization, the entity-specific websites 22 are created for each entity contracting with the photo-sharing service 16 in step 54. Customization requires storage of entity information in the entity/account database 20 and creating and storing web page elements comprising the entity-specific photo-sharing website in the database 20. The entity-specific information stored in the database 20 may also include service levels, and enabled features for the entity-specific website 22. Features are components or services that may be provided on websites by the photo-sharing service 16, such as search functions, and online printing, for instance, but may be selectable by each entity for its own website. As an example, company X may provide customized cameras 14 for its employees, but may not wish to allow employees to print images from the company X photo website for security reasons. If so desired, company X may have the photo service disable this feature from their particular website.

In a preferred embodiment, the entity-specific websites 22 are not created from scratch, but are created by modifying a preexisting template. The template may include several different sections, such as A, B, C and D, for instance. Assuming for example that the template used to create a website for Nikon, and section A is used to specify the name of the entity then the name Nikon would be inserted into that section. Other entity-specific content would be used to fill out the remaining sections. The Web pages comprising the Nikon specific photo-sharing website would then be provided with URL's unique to that website. The entity's regular website would be modified to include a link to the entity's photo-sharing website 22. In addition, the entity-specific photo-sharing website would include a link back to the entity's website. Entities 12 may have entity photo-sharing websites 22 created for them in one of two ways; automatically by logging into the photo-sharing service 16 and manually customizing the templates, or by having the entity photo-sharing website created for them.

Referring still to FIG. 2, when a camera 14 establishes an internet connection with the gateway server 18, the camera 14 transmits its entity IDs 28 and/or user entity ID 28 when uploading user selected images to the gateway server 18 in step 56. In response, the gateway server 18 matches the entity ID 28 with an entity account in the database 20 and

8

matches the user entity ID 28 with a user account 40 in the database 20 in step 58. The images received are then stored in the user account 40 of the identified entity-specific website 22 in step 60.

Referring again to FIG. 1, each user account 40 in the database 20 may also include one or more action lists 48. According to the present invention, an action list 48 includes one or more items representing actions that the gateway server 18 should take with respect to uploaded images, such as where to store and/or send the images from a particular user or camera, for instance. As explained further below, the action list 48 stored on the database 20 under a user's account 40 are automatically downloaded to the user's camera 14 during a connection with the gateway server 18 and stored on the camera 14. When the user initiates an image upload, the action list 48 is displayed to the user so the user may easily select what actions the gateway server 18 should take with respect to the images by selecting the displayed action list items.

Examples of action list items include specifying that the uploaded images should be stored on the entity-specific photo website, sending the images to a list of email addresses, or even performing some type of analysis or calculation on the image data, for instance.

In a further aspect of the present invention, an action list item is not limited to, instructing the gateway server 18 to perform actions only within the photo-sharing service 16. Rather, an item in the action list 48 may also instruct the gateway server 18 to perform actions outside of the photo-sharing service 16, such as storing the images in an external database 49 of the entity 12. For instance, in the example where the entity 12 is a company, some users of the company's cameras 14 could have action lists 48 instructing the gateway server 18 to store uploaded images to the company's database, rather than to the company's photo-sharing site 22. Based on the action lists 48 and customization, the gateway server 18 may be programmed to automatically perform predefined tasks, such as creating new web albums, or a new page within an existing album, parse the images to extract sound files or other metadata, print images and mail them to designated addresses, and so on.

In a preferred embodiment, the action lists 48 may be created via several methods. In one method, the action list is created by the photo-sharing service 16 the first time the user's camera establishes a connection. That is, a default action list 48 is automatically created based on the entity ID when a user account 40 is first created. In a hierarchical entity relationship where the entity 12 is a company, a default action list 48 may be created to implement a workflow specified by the entity 12. In a hierarchical relationship where the entity 12 is camera manufacturer, for instance, a default action list 48 may be created instructing the gateway server 18 to store the user's images in a simulated "shoebox" on the entity-specific photo-sharing site 20. The user may then go online and create albums from the images in the shoebox as desired.

Another method is for the user to create the action list 48 online on the entity-specific photo-sharing site 20. The action list 48 may be created manually on the website 20 by the user navigating to the site 20 using a web browser 24, accessing her account, and manually creating the action list 48 or editing the action list 48 on the entity-specific site 22. The action list 48 may also be created automatically on the website 20 in response to user actions performed on the website, such as printing images, or creating a web album.

9

Alternatively, after performing an action, the user may be prompted whether they would like this action added to his or her action list 48. If so, the user clicks a check-box and the item is added the action list 48. In a preferred embodiment, any action list 48 created and edited on the photo-sharing site 20 are downloaded to the camera every time the camera 14 connects to the photo-sharing service 16 and made available for user selection on the camera 14 during the next upload.

Yet another method for creating an action list 48 is to allow the user to create the action list on the camera 14. The user may manually create an action list 48 by "typing" in predefined items on the camera. The user may also type in an email address as an action list item whereby when that item is selected, the uploaded images are stored as a web album on the entity-specific photo-sharing website 22 and the server 18 sends a notification to the specified recipient containing the URL to the web album page.

A method and system for hosting web-based photo-sharing websites and for customizing digital cameras to upload images to the entity-specific photo-sharing websites has been disclosed. According to the present invention, users of the customized cameras 14 can upload images to the Internet for storage and web photo album creation without the use of a PC.

In one embodiment described above, the present invention assumes that an ISP account has been established with the digital camera's service provider, and that users of cameras 14 belonging to a certain entity 12 may use the cameras 14 to upload images to the website of the entity 12. However, two problems with account setup remain. One problem is that just as with a PC and PDA, the user must first establish an ISP account before the camera 14 can establish Internet communication. The second account problem is that most websites, including the photo sharing sites 22, may require each user to establish a unique account before using the site to distinguish one user from another. Before being able to connect the web-enabled camera 14 to the Internet, the user must establish these two accounts by either entering account setup information on a PC or entering account setup information on the camera 14. Neither alternative is a convenient alternative for people who do not have the time nor inclination to do so.

In a further aspect of the present invention, the cameras and the photo sharing site are provided with software for automatically creating Internet and photo-sharing website accounts for each camera 14 upon first use, without requiring the user to first enter account information on a PC or on the camera 14.

Referring now to FIGS. 4A and 4B, a flow chart illustrating a process for automatically configuring a web-enabled digital camera to access the Internet in a preferred embodiment of the present invention. Although the process will be described in terms of automatically establishing Internet accounts for a digital camera without requiring the user to enter information, those with ordinary skill in the art will readily recognize that the present invention may be used to automatically establish Internet accounts for any type of portable electronic device.

The process assumes that a user has just acquired a digital camera 14 customized as described above, and has just taken the camera 14 out of its box in step 100. After taking pictures with the camera in step 102, the user may review the images in the camera's LCD screen and optionally select a set of images to upload to the photo sharing service 16 in step 104. The user then presses a "send" button on the camera in step 106 to upload the images.

10

In response, the camera displays a confirmation dialog screen on the camera and prompts the user to record an audio annotation for the images or to continue in step 108. The user may then choose to record audio for the images in step 110. After choosing to continue, or after recording audio, the camera displays a "connecting" dialog screen in step 112 to indicate to the user that the camera is establishing an Internet connection. At the same time, the camera checks for available connection options in step 114, and if more than one is found, the camera prompts the user to select one of the connection options. For example, the camera may be within range of a Bluetooth-equipped printer and a cellphone, so the user will be prompted to choose which device the camera should establish communication with.

The camera then checks for the presence of account information on the camera in step 116, and if there are none, the camera assumes that this is a first-time connection. According to the present invention, in order to allow the camera to make a first-time Internet connection, the camera is provided with default Internet service provider (ISP) information during manufacturing, including an ISP access number, and user ID and password (if required). The camera establishes connection with the default ISP in step 118 by dialing the preloaded access number, and by sending the preloaded user ID and password to the ISP. This special account may be configured so that the camera can only connect to the gateway server 18 (no other IP addresses may be allowed).

After connecting with the ISP, the camera connects to the gateway server 18 and sends unique camera information and/or user information in step 120. In a preferred embodiment of the present invention, a combination of the camera's serial number and the make and model number of the camera may be sent as the unique camera information. In another preferred embodiment, the user's e-mail address may be sent as the unique camera or user information.

Continuing with FIG. 4B, the gateway server 18 uses the unique camera information to set up a user account 40 in step 122. After creating the user account 40, the gateway server 18 returns an information packet to the camera containing new ISP information (if needed), an account ID, and an account password in step 124. The information packet may also contain a default action list specifying what actions should be taken with respect to the images, an advertisement for display on the camera, and the URL of the entity-specific website 22.

It should be noted that if the camera is used in conjunction with an IP direct phone or is provided with a phone number for connected to a dedicated server where the user is not billed separately for the ISP connection, then the steps of providing the camera with default ISP info and returning new ISP info, may be omitted.

The camera receives the information packet in step 126, and writes a configuration file to memory 82a containing the ID, password, and default action list in step 128. The camera then displays the action list on the camera's LCD screen for selection by the user in step 130. The camera may optionally display the user's account information as well.

In an alternative preferred embodiment where security is a concern, the camera 14 first logs off the special ISP and the gateway accounts, and then reconnects using new ISP and gateway accounts in order to retrieve the action lists 48.

FIG. 5 is a block diagram illustrating an example action list 48 displayed on the LCD screen of the camera 14. The action list 48 is shown displaying three major options; printing the uploaded images, saving the uploaded images in

11

the user's shoebox, and sending the images to Mom. Under the printing option, the user may select from various size prints. Rather than nested menu categories as shown under the printing option, the action list 48 may be displayed with each action listed as a separate item (e.g., "Send 4x8 prints to Mom", "Send 5x7 prints to me").

Referring again to FIG. 4B, after the user selects one or more actions from the action list 48, the camera begins to upload the images along with the selected actions in step 132 and displays a progress bar on the screen in step 134. In one preferred embodiment, the camera may also display the advertisement sent in the information packet from the gateway server 18. The advertisement may advertise the controlling entity 12, the entity's photo-sharing site 22, or the photo sharing service 16. After all the images are uploaded and associated with the user's account 40, the camera breaks the connection with the gateway server in step 136. At this point, the camera 14 may also display the URL of the entity-specific website 22 to the user.

The next time the user uploads images, the camera will use any new ISP information received to connect to the Internet and will use the account ID and password written to memory 82a when connecting to the gateway server 18. Thus, by using unique camera information, such as the serial number, to establish a web site account upon first use, the present invention eliminates the need for the user to type in information to establish a web site accounts.

To further explain the present invention from a user interaction point of view, consider the following scenario where a user named Jack has just purchased a digital camera 14 from a store and unpacks the camera from the box. There is a "Quick Start Guide" which guides him in getting started. Jack pops in the batteries, sets the date and time, and takes some pictures of his dog and his new baby. Jack can see the pictures have come out well on the small LCD, and now wants to try sharing them with his parents.

The Quick Start Guide says to select the photos to be sent using the "Select" button, and then press the "Send" button. So, Jack navigates to each of the baby pictures, selects them, and then presses the "Send" button. Instantly, a dialog comes up on the LCD screen: "No Receiving Device Found! Please turn on your phone or other connecting device." Oops! Jack pulls out his cell phone, and turns it on. He presses the "Continue" button. Jack does not see this happen, but the camera now "discovers" the cell phone, and immediately presents another dialog: "4 Images Selected. Press Record to add a Sound Note, or Continue to send". Although Jack finds the proposition of recording sound intriguing, he decides to skip it and presses the "Continue" button. Immediately, the dialog is replaced with a "Connecting . . ." dialog.

Shortly, another dialog appears: "Your camera serial number is 38147. Please write this down. You will need it to access your web photo albums". Jack writes the number down on the spot provided in the Quick Start Guide, and presses OK.

Another "Connecting . . ." dialog appears, and is then quickly replaced by another dialog, which says "A free, temporary account has been set up for you at www.photosharing.service.com/new_accounts. You will need your camera serial number to access your photos and complete the setup of your account. Please complete the account setup within 30 days". Jack writes down the URL in the space provided in the Quick Start Guide, and presses the OK button. Jack doesn't know it, but during this dialog, the camera has begun transmitting his images and is already partially complete.

12

A new dialog comes up, with a progress bar. Jack is surprised to see that the transmission is already almost ½ done. Below the progress bar, the dialog says "Press 'Continue' to use camera during photo transmission, or wait for progress bar to complete". Jack is interested in watching how fast his images are transmitted, and decides to watch the progress bar complete. A "Transmission Successful" dialog appears. Jack presses the OK button. The camera returns to the review mode.

Jack is pretty excited—he just sent four baby pictures to the Internet. Jack then decides to see what happened to his images so he turns on his PC. After connecting to the Internet, Jack types in the URL from the Quick Start Guide. A Photo-sharing service web page appears, welcoming Jack to the photo-sharing service. After looking briefly at the welcome page, Jack types in his serial number from the Quick Start Guide, and selects his camera's model number from a pop-up menu. Jack clicks on a "Submit" button on the web page.

Jack now sees a page which shows thumbnails of the baby pictures he just sent, the page is entitled "My Shoebox". The page explains to Jack that he is looking at his on-line digital photo shoebox. Since the server 18 knows this is Jack's first visit, special help messages may appear. Various options are provided via buttons and text links. One that catches Jack's eye is CREATE WEB PHOTO ALBUM. Jack clicks this button, and works his way through the process of setting up an on-line photo album. This includes selecting photos from the shoebox, as well as selecting layout and style. One of the check-box items Jack is offered is "Make this album a camera Action List Item". Jack doesn't know what that is, so he clicks the Action List link, which brings up a brief description "If you check this box, you will be able to send pictures directly from the camera to this photo album!" Jack finds this interesting, so he closes the description window, and checks the box. Jack also enters the email address for his parents and his wife's parents, so they can be notified to come and see his photo album, which he has entitled "Our First Baby".

One of the buttons Jack does not click is the "Complete Account Setup" button. He knows that he has 30 days to do that chore, and figures he will get back to it later.

Jack's wife arrives home from shopping at this point, and Jack wants to show her how the new camera works. He starts by showing her the baby album on the PC, then decides to take some pictures of them holding the baby. Jack then selects the images, and presses the "Send" button again.

Since his cell phone is still on, sitting on the table a few feet away, the connection goes smoothly and quickly. A new dialog pops up, surprising Jack. It says "Select the destination for your pictures" and offers two choices: "My Shoebox" and "Our First Baby". Jack is amazed, he doesn't realize that the server 18 downloaded his action list to his camera during the connection. Jack decides to select the "Our First Baby" web album as the destination for the images, and clicks OK. The pictures are sent as before. After the transmission has completed, Jack goes to the PC to check on the photo album. When Jack refreshes the album page, he now sees the additional pictures he just sent, along with the pictures he sent before.

Jack spots a "Send Prints" link on the web page, and clicks it. He is led through a selection of print types, mailing addresses, and credit card info to make it possible to send prints. He is offered the option of completing the setup of his account. Jack decides to do that now, and proceeds to fill out the requested information, including his credit card number.

13

Once the account setup is complete, Jack continues the print order. One of the radio button items is "Make prints a separate Action List Item" or "Make prints part of your Action List". Jack remembers something about Action Lists from before, but is not sure what this means. The description says "Making a separate action list for prints allows you to decide in the camera to send to the photo album and send prints or to just send to the photo album." Jack thinks this is cool, and clicks the "Make prints a separate Action List" item. The next time Jack sends photos from the camera, his action list will be updated to present three choices: My Shoebox, Our First Baby, and Our First Baby w/Prints.

The underlying technology supporting this scenario are summarized below. Other functions and features are assumed, but not required for this scenario:

1. Two-way connection between camera and portal
2. Camera metadata included in the request
3. If not using an IP direct connection,
 - 3.1. Default ISP connection info built into the camera for the first connection (country specific)
 - 3.2. Downloaded assigned ISP information from the portal to the camera
4. Software capable of recognizing automatically a set of supported phones and adjusting the protocol to match
5. Action Lists maintained on the server that are automatically downloaded to the camera to update the camera selection list each time a connection is made
6. An On-line shoebox
7. Ability to download files to the camera from the server. The files could be text, GIF, animated GIF, JPG, or even a script or applet. This feature enables the ability to display advertisements on the camera, remind the user of remaining time to complete account setup, make special offers, and indicate limits reached.

A method and system for automatically configuring a web-enabled digital camera to access the Internet has been disclosed. Although the present invention has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations to the embodiments and those variations would be within the spirit and scope of the present invention. For example, although the photo-sharing service has been described as including the gateway server and the database, the database may be located elsewhere. Also, the gateway server may be used to control account information, while one or more other servers may be used to provide the web pages of the entity-specific websites. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.

What is claimed is:

1. A method for automatically establishing a user account and for configuring a hand-held image capture device which includes a digital camera for accessing a site on a public network using the user account, the method comprising the steps of:

- (a) establishing a connection to a website server, the website server being responsible for establishing and maintaining website accounts;
- (b) checking for the presence of account information on the image Capture Device and if none are found, determining that this is a first-time connection and automatically sending image capture device information uniquely identifying the electronic device to the website server so that the server can set-up the account information based on the image capture device information;

14

(c) receiving user account information from the server, wherein the user account information includes an account ID and password; and

(d) storing the user account information on image capture device for use a next time the image capture device accesses the website, whereby the user does not have to enter account information in order to establish the connection with the public network or to establish the website account.

2. The method of claim 1 further including the step of providing a digital camera as the hand-held image capture device.

3. The method of claim 2 further including the step of sending at least a partial serial number as the image capture device information uniquely identifying the electronic device.

4. The method of claim 3 further including the step of uploading captured images to the website server.

5. The method of claim 4 wherein step (a) further including the steps of:

- (i) providing the image capture device with default ISP information; and
- (ii) establishing a connection to an ISP using the default ISP information.

6. The method of claim 5 further including the step of terminating the connection with the ISP when the uploading of the images is complete.

7. The method of claim 6 further including the step of providing the digital camera with a default ISP access number, user ID, and password.

8. A system for automatically establishing a user account and for configuring a hand-held image capture device which includes a digital camera image capture device for accessing a site on a public network using the user account, comprising:

means for establishing a connection to a website server, the website server being responsible for establishing and maintaining website accounts;

means for checking for the presence of account information on the image capture device, and if none are found, determining that this is a first-time connection and automatically sending image capture device information uniquely identifying the image capture device to the website server so that the server can set-up the account information based on the image capture device information; and

means for receiving user account information from the server, wherein the user account information includes an account ID; and

means for storing the user account information on image capture device for use a next time the image capture device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account.

9. The system of claim 8 wherein the hand-held image capture device comprises a digital camera.

10. The system of claim 9 wherein the image capture device information uniquely identifying the image capture device includes a serial number of the image capture device.

11. The system of claim 10 wherein the camera uploads captured images to the website server.

12. The system of claim 9 wherein the camera further includes means for providing the image capture device with default ISP information, and means for establishing a connection to an ISP using the default ISP information.

13. The system of claim 12 wherein the digital camera terminates the connection with the ISP upon completion of image uploading.

15

14. The system of claim 13 wherein the camera is provided with a default ISP access number, user ID, and password.

15. A method for automatically establishing a user account and for configuring a digital camera to access a website on a public network using the user account, the digital camera including captured images and communication means for accessing the public network, the method comprising the steps of:

- (e) storing default ISP information on the digital camera, the default ISP information including an access number;
- (f) in response to a user request to upload images to the website, determining whether this is a first-time connection;
- (g) if it is first-time connection, establishing communication with the default ISP by dialing the access number;
- (h) establishing communication with the website and automatically sending a digital camera ID to the website;
- (i) using the digital camera ID to set up a user account on the website;
- (j) sending user account information to the digital camera, the user account including an account ID and an account password;
- (k) storing the user account information on the camera; and
- (l) uploading images to the website and associating the images with the user account, whereby the user does not have to enter information to establish a connection with the public network or to establish the website account.

16. The method of claim 15 further including the step of sending at least a partial serial number as the digital camera ID.

17. The method of claim 16 further including the step of receiving new ISP information from the website.

18. The method of claim 17 further including the step of using the new ISP information a next time the digital camera establishes a connection with the public network.

16

19. A system for automatically establishing a user account and for configuring a digital camera for accessing a website on a public network using the user account, the digital camera including images captured by user and communication means for wirelessly accessing the public network, the system comprising the steps of:

- means for storing default ISP information on the digital camera, the default ISP information including an access number;
- means for determining whether this is a first-time connection in response to a user request to upload images to the website;
- means for the establishing communication with the default ISP by dialing the access number;
- means for establishing communication with the website and sending a digital camera ID to the website;
- means for using the digital camera ID to set up a user account on the website;
- means for automatically sending user account information to the digital camera, the user account including an account ID and an account password;
- means for storing the user account information on the camera; and
- means for uploading images to the website and associating the images with the user account, whereby the user does not have to enter information to establish a connection with the public network or to establish the website account.

20. The system of claim 19 wherein the digital camera ID includes a serial number.

21. The system of claim 20 wherein the digital camera receives new ISP information from the website.

22. The system of claim 21 wherein the digital camera uses the new ISP information a next time the digital camera establishes a connection with the public network.

* * * * *



US00635384B1

(12) **United States Patent**
Morris

(10) **Patent No.:** **US 6,353,848 B1**
(45) **Date of Patent:** **Mar. 5, 2002**

(54) **METHOD AND SYSTEM ALLOWING A CLIENT COMPUTER TO ACCESS A PORTABLE DIGITAL IMAGE CAPTURE UNIT OVER A NETWORK**

(75) Inventor: **Robert P. Morris**, Raleigh, NC (US)

(73) Assignee: **FlashPoint Technology, Inc.**,
Peterborough, NH (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/127,514**

(22) Filed: **Jul. 31, 1998**

(51) Int. Cl.⁷ **G06F 15/16**

(52) U.S. Cl. **709/203; 709/200; 709/202; 709/217; 709/219; 709/238; 709/232; 709/245**

(58) **Field of Search** **709/200-203, 709/206, 217-219, 227-229, 231-232, 236-237, 245; 707/10, 200-204; 713/201-202**

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,911,044 A	~	6/1999	Lo et al.	709/203
6,018,774 A	~	1/2000	Mayle et al.	709/203
6,058,428 A	~	5/2000	Wang et al.	709/232
6,085,249 A	~	7/2000	Wang et al.	709/229
6,101,536 A	~	8/2000	Kolani et al.	709/217
6,141,759 A	~	10/2000	Braddy	709/203

FOREIGN PATENT DOCUMENTS

DE 198 08 616 9/1998 H04L/12/16

OTHER PUBLICATIONS

De Albuquerque et al., "Remote Monitoring Over The Internet", Nuclear Instruments & Methods In Physics Research, Section-A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 412, No. 1, Jul. 21, 1998, pp. 140-145.

* cited by examiner

Primary Examiner—Zarni Maung

Assistant Examiner—Bharat Barot

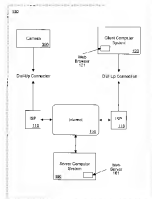
(74) Attorney, Agent, or Firm—Sawyer Law Group LLP

(57)

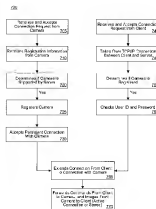
ABSTRACT

A method for accessing a digital image capture unit via a communication network comprising a server computer system and a client computer system communicatively coupled with communication equipment. In one embodiment, the address of the digital image capture unit is registered in an executable program on the server computer system. The executable program is accessed by the client computer system. The executable program connects the digital image capture unit and the server computer system. The executable program communicates commands between the client computer system and the digital image capture unit, such that data captured by the digital image capture unit is transferred to the client computer system via the server computer system.

27 Claims, 11 Drawing Sheets



126



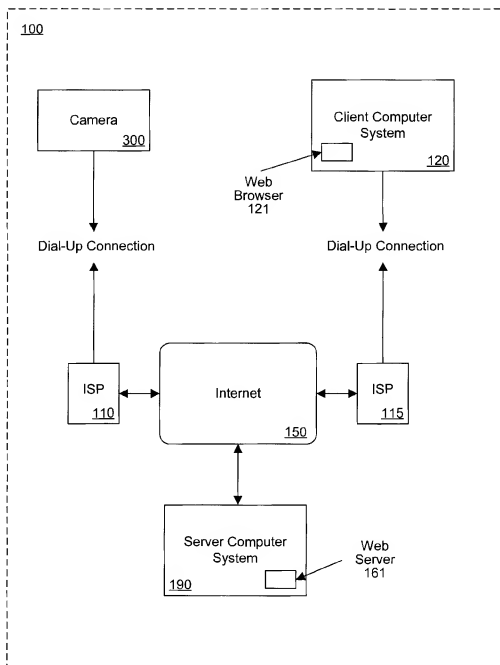


FIG. 1A

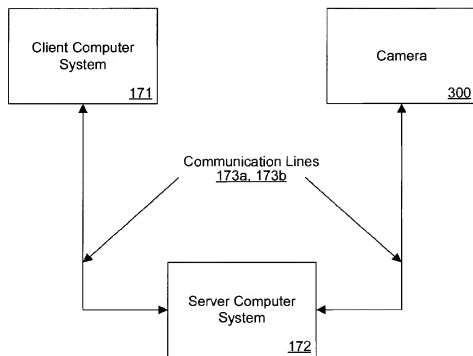
170

FIG. 1B

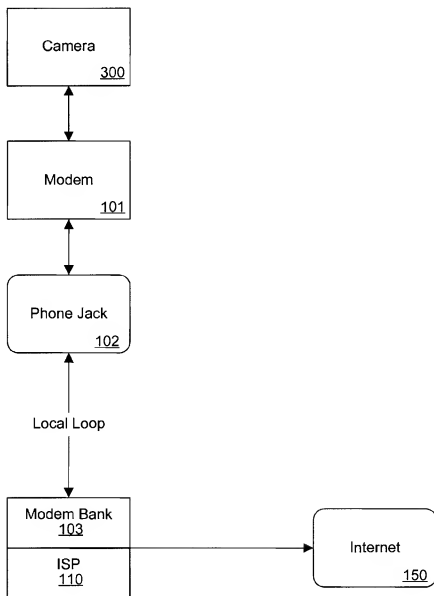


FIG. 1C

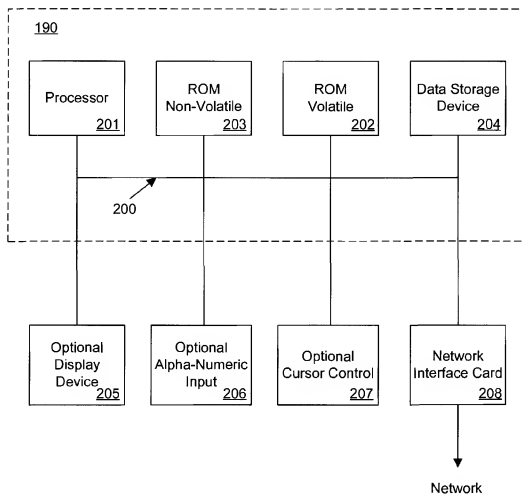


FIG. 2

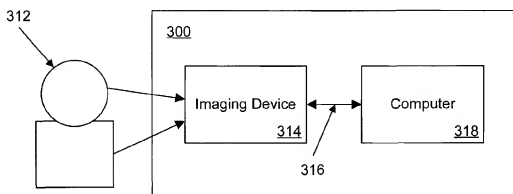


FIG. 3

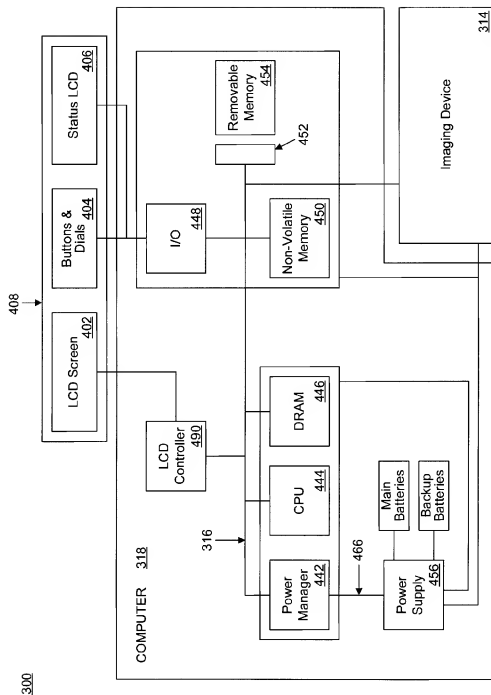


FIG. 4

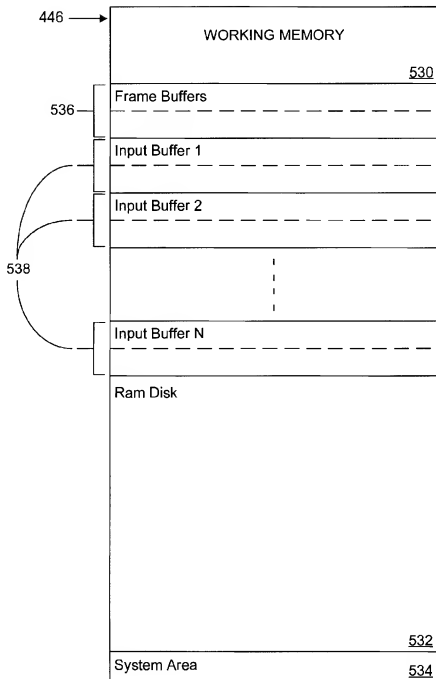


FIG. 5

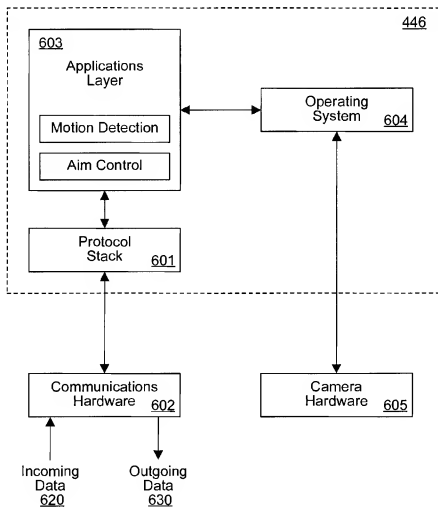


FIG. 6

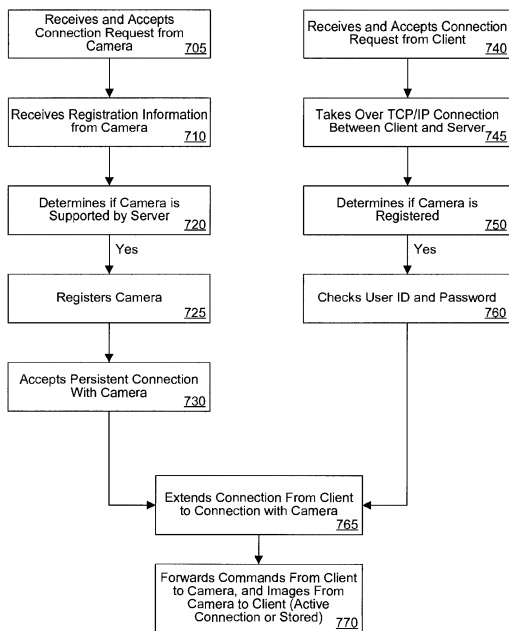
700

FIG. 7

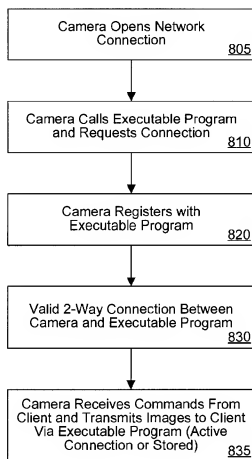
800

FIG. 8

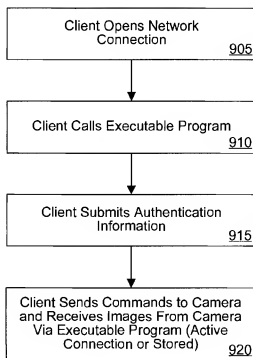
900

FIG. 9

METHOD AND SYSTEM ALLOWING A CLIENT COMPUTER TO ACCESS A PORTABLE DIGITAL IMAGE CAPTURE UNIT OVER A NETWORK

FIELD OF THE INVENTION

The field of the present invention pertains to digital image capture devices. More particularly, the present invention relates to a method for remotely accessing a digital camera via a communication network.

BACKGROUND OF THE INVENTION

Modern digital cameras typically include an imaging device which is controlled by a computer system running a software program. When an image is captured, the imaging device is exposed to light and generates raw image data representing the image. The raw image data are typically stored in an image buffer, where they are processed and compressed by the computer system's processor. Many types of compression schemes can be used to compress the image data, such as the joint photographic expert group (JPEG) standard. After the processor processes and compresses the raw image data into image files, the processor stores the image files in an internal memory or on an external memory card.

Some digital cameras are also equipped with a liquid-crystal display (LCD) or another type of display screen on the back of the camera. Through the use of the LCD, the processor can cause the digital camera to operate in one of two modes, play and record, although some cameras only have a record mode. In the play mode, the LCD is used as a playback screen allowing the user to review previously captured images either individually or in arrays of four, nine, or sixteen images. In the record mode, the LCD is used as a viewfinder through which the user may view an object or scene before taking a picture.

Besides the LCD, user interfaces for digital cameras also include a number of buttons or switches for setting the camera into one of the two modes and for navigating between images in play mode. For example, most digital cameras include two buttons, labeled "-", and "+", that enable a user to navigate or scroll through captured images. For example, if the user is reviewing images individually, then pressing one of navigation buttons causes the currently displayed image to be replaced by the next image.

A digital camera has no film and, as such, there is no incremental cost of taking and storing pictures. Hence, it is possible to take an unlimited number of pictures, wherein the most recent picture replaces the earliest picture, for virtually zero incremental cost. Accordingly, this advantage is best realized when the camera is used as much as possible, taking pictures of practically anything of interest.

One way to best utilize this unique attribute is to make the digital camera and its internally stored images remotely accessible. If the pictures are remotely accessible, the camera could be set to continuously take pictures of scenes and items of interest. Ideally, a user would be able to access those pictures at any time. The user would be able to use a widely available communications medium to access the camera from virtually an unlimited number of locations.

The emergence of the Internet as a distributed, widely accessible communications medium provides a convenient avenue for implementing remote accessibility. Providing remote accessibility via the Internet leverages the fact that the Internet is becoming familiar to an increasing number of

people. Many users have become accustomed to retrieving information from remotely located systems via the Internet. There are many and varied applications which presently use the Internet to provide remote access or remote connectivity. Internet telephony is one such application, such as Microsoft's NetMeeting and Netscape's CoolTalk.

NetMeeting and CoolTalk are both real-time desktop audio conferencing and data collaboration software applications specifically designed to use the Internet as their communications medium. Both software applications allow a "local" user to place a "call" to a "remote" user located anywhere in the world. With both NetMeeting and CoolTalk, the software application is hosted on a personal computer system at the user's location and on a personal computer system at the remote user's location. Both NetMeeting and CoolTalk require a SLIP (Serial Line Internet Protocol) or PPP (Point-to-Point Protocol) account where Internet access is via a dial-up modem, and where the user, as is typical, accesses the Internet through an ISP (Internet Service Provider). Both NetMeeting and CoolTalk require personal computer systems for the resources necessary to run these applications (e.g., processing power, memory, communications hardware, and the like). In addition, both NetMeeting and CoolTalk require the one user to input an IP (Internet Protocol) address for the other user in order to establish communication between the users.

To facilitate the process of obtaining appropriate Internet addresses, CoolTalk, for example, allows on-line users to list their respective IP addresses with a proprietary CoolTalk central Web server. This allows a user to obtain a list of users currently on-line to whom communication can be established. Upon locating the desired remote user in the Internet address list maintained by the Web server, the local user places the call.

In this manner, the proprietary CoolTalk Web server maintains a user-viewable and user-updated "address book" in which users list their respective Internet addresses and in which they search for the Internet addresses of others with whom they wish to communicate. However, both NetMeeting and CoolTalk require active user input, in that each require the user to input his current Internet address, and in that each require the local user to search the address book for the Internet address of the remote user to be contacted. This can be quite problematic in the case where users obtain access to the Internet via dial-up connections and hence have different Internet addresses each time their respective dial-up connections are established.

In a manner similar to Internet telephony, Internet desktop video conferencing is another prior art application which uses the Internet as its communications medium. One such application, for example, is CU-SeeMe by White Pine. CU-SeeMe provides real time video conferencing between two or more users. As with NetMeeting and CoolTalk, CU-SeeMe is a software application which runs on both the local user's personal computer system and the remote user's personal computer system. The personal computer systems provide the resources for running the application. As with NetMeeting and CoolTalk, CU-SeeMe requires the local user to enter the IP address of the remote user. Like CoolTalk, CU-SeeMe facilitates this process by allowing on-line users to list their respective IP addresses with a proprietary central Web server such that the addresses can be easily indexed and searched.

Another prior art example of remote access via the Internet is status queries of remote devices using the Internet as the communications medium. A typical prior art applica-

3

tion involves interfacing a remote device with a computer system, and accessing the computer system via the Internet. For example, a vending machine can be remotely accessed to determine its status (e.g., the number of sales made, whether the machine needs refills, whether the machine needs maintenance, and the like). The machine is appropriately equipped with sensors, switches, and the like, which in turn are interfaced to a computer system using a software driver. The computer system is coupled to the Internet and interfaces with the machine through the driver, making the relevant information available over the Internet using Web server software. Hence, any interested user (e.g., the vending machine service company) is able to remotely ascertain the status of the machine via the Internet.

A problem with the above described prior art applications is that access to the Internet and communication thereon require a separate host computer system (e.g., a personal computer system) on each side of the Internet connection in addition to the server computer system on the Internet. The two host computer systems provide the computational resources to host the respective software applications, the Internet access software, and any necessary device drivers. The required computational resources consume a significant amount of memory. Because of this, among other reasons, the above prior art applications are not easily transferred to the realm of easy-to-use, intuitive, consumer electronic devices such as digital cameras, which are small in size and so generally constrained by the amount of memory they can house. In addition, a consumer electronic device such as a digital camera that requires a separate computer system would be more expensive and complex, and therefore would not be consistent with the desire of consumers for lower cost and simpler devices.

Also, separate host computer systems (where the host computer systems host the software and drivers required by prior art applications as described above) require extra effort to administer, particularly with regard to networks consisting of a large number of computer systems (e.g., digital cameras each incorporating a computer system). For example, an upgrade to the software residing on each computer system has to be individually installed on each computer system. Also, each computer system has to be individually polled to query whether the computer system has data of interest to the user, and then the data have to be separately accessed and collected from each computer system, then compiled. For example, in an application involving digital cameras, a user may be interested in finding out which digital cameras have images in storage. In the prior art, the user has to access each digital camera individually. In another case, a user may have an interest in maintaining a record of transactions between all users and all digital cameras. Again, in the prior art this is accomplished by individually accessing each digital camera (or, alternatively, each user's computer system) to collect the data, and then compiling the complete list of transactions.

Another problem with the prior art is the fact that the applications described above require the user to know the Internet address of the person or device that is being contacted. The Internet telephony applications (e.g., CoolTalk) often employ a user-viewable and user-updated address book to facilitate the process of locating and obtaining the correct Internet address; however, they require active user input. This is difficult in the case where users obtain access to the Internet via dial-up connections, and thus have changing Internet addresses. Still another problem with the prior art is that the applications described above provide only a limited degree of functionality; that is, they are

4

limited to either chat, video conferencing, or the like. As such, they are not capable of establishing a connection between any type of user system and remote device.

One prior art system is described in the copending previously filed patent application, assigned to the assignee of the present invention, entitled "A Method and System for Hosting an Internet Web Site on a Digital Camera," Eric C. Anderson and others, Ser. No. 09/044,644. This prior art system presents one solution to the problem of gaining remote access to those digital devices where the location and Internet address of the device are highly changeable. This prior art system incorporates a Web server into the digital device, specifically a digital camera. However, the disadvantage to this prior art system is that the Web server consumes valuable memory and computational resources in the digital camera. In addition, because of the limited memory in a device such as a digital camera, the Web server is not as powerful as a Web server on a server computer system.

Thus, a need exists for an inexpensive and powerful method for implementing remote access to digital devices, such as digital cameras, where the location and Internet address of the device are highly changeable. A further need exists for an intuitive, simple protocol for presenting the device's functionality and capabilities to users. In addition, a need exists for a method of efficiently administering a plurality of separate devices. A need also exists for an efficient process of obtaining the address of the device that is transparent to the viewer. The present invention provides a novel solution to the above needs.

SUMMARY OF THE INVENTION

The present invention provides an inexpensive and powerful method for implementing remote access to digital devices, such as digital cameras, where the location and Internet address of the device are highly changeable. The present invention further provides an intuitive, simple protocol for presenting the device's functionality and capabilities to users. In addition, the present invention provides a method of efficiently administering a plurality of separate devices. The present invention also provides an efficient process of obtaining the address of a device that is transparent to the user.

In one embodiment, the present invention is an executable program for accessing a digital camera via a communication network using a Web server on a server computer system and a Web browser (or a program of similar function) on a client computer system that are communicatively coupled via the Internet. The address of the digital camera is registered in an executable program on the server computer system. The executable program is accessed by the client computer system. The executable program connects the digital camera and the server computer system. The executable program enables the client computer system and the digital camera to communicate using any protocol used by these devices, thus allowing data (e.g., images) acquired by the digital camera to be transferred to the client computer system.

The executable program can be implemented in a variety of forms. For example, the executable program can be a Java script. Alternatively, the executable program can be a cgi-bin (Common Gateway Interface-binaries).

For example, in the case of a digital camera, the executable program directly communicates commands from the client computer system to the digital camera when both the client computer system and the digital camera are on-line at the same time. Alternatively, if the digital camera is not

5

on-line, the commands from the client computer system are first stored in the server computer system, and then later communicated by the executable program to the digital camera after a connection between the server computer system and the digital camera is made. The capability to store and then forward commands and data is not limited to a digital camera application nor is it limited to a particular data storage format. The data storage format can be any format that is understood by both the client computer system and the digital device.

Images and any other data acquired by the digital camera are accessed by the server computer system using the executable program and directly transferred to the client computer system when both the client computer system and the digital camera are on-line at the same time. Alternatively, if the client computer system is not on-line, the data are first stored by the server computer system, and then later communicated to the client computer system after a connection between the server computer system and the client computer system is made.

It should be noted, however, that the present invention can be readily modified to function in other embodiments, such as, for example, hand-held digital devices, lap top personal computers, and the like, which require an efficient process of obtaining the address of a device that is transparent to the user.

These and other objects and advantages of the present invention will become obvious to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments which are illustrated in the various drawing figures.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not by way of limitation, in the figures of the accompanying drawings in which like reference numerals refer to similar elements and in which:

FIG. 1A shows a block diagram of a remote access system via the Internet in accordance with one embodiment of the present invention.

FIG. 1B shows a block diagram of a remote access system via a Local Area Network in accordance with one embodiment of the present invention.

FIG. 1C shows a block diagram of a digital camera coupled to the Internet via an Internet Service Provider.

FIG. 2 shows a general purpose computer system upon which embodiments of the present invention may be practiced.

FIG. 3 shows a block diagram of a digital camera for use in accordance with the present invention.

FIG. 4 shows a block diagram of a computer system of a digital camera in accordance with one preferred embodiment of the present invention.

FIG. 5 shows a memory map of a dynamic random access memory of a digital camera in accordance with one embodiment of the present invention.

FIG. 6 shows a diagram of the connectivity and application software of a digital camera in accordance with one embodiment of the present invention.

FIG. 7 is a flowchart of a process for remotely accessing a digital camera implemented by an executable program in accordance with one embodiment of the present invention.

FIG. 8 is a flowchart of a process employed by a digital camera for remote access in accordance with one embodiment of the present invention.

6

FIG. 9 is a flowchart of a process employed by a client computer system for remote access of a digital camera in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In the following detailed description of the present invention, numerous specific details are set forth in order to enable one of ordinary skill in the art to make and use the invention, and are provided in the context of a patent application and its requirements. Although the present invention will be described in the context of a digital camera, various modifications to the preferred embodiment will be readily apparent to those skilled in the art, and the generic principles herein may be applied to other embodiments. That is, any digital device which displays data, images, icons and/or other items, could incorporate the features described below and that device would be within the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

Some portions of the detailed descriptions which follow are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computer memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is here, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computer system. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, bytes, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that throughout the present invention, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes (e.g., the processes of FIGS. 7, 8 and 9) of a computer system or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present invention provides a method for making a digital device (e.g., a digital camera) and its internally stored data remotely accessible over a communication network such as the Internet or a Local Area Network (LAN). The present invention is an executable program placed on a server computer system (specifically, a Web server) that implements commands initiated by a client (or user) using a client computer system with a Web browser or a program of similar function. The present invention enables the digital camera to be set to continuously take pictures of scenes/

7

items of interest and allow a client to access those pictures at any time. The present invention allows the client to access the digital camera from virtually an unlimited number of locations and with the camera in virtually any location.

Referring now to FIG. 1A, a block diagram of communication network 100 is shown. Communication network 100 provides a method in accordance with one embodiment of the present invention which implements remote access to camera 300 and its internally stored data. Communication network 100 includes camera 300, Internet Service Provider (ISP) 110, Internet Service Provider 115, client (or user) computer system 120, and server computer system 190. ISP 110 and ISP 115 are both directly coupled to the Internet 150. Client computer system 120 includes Web browser 121 or a program of similar function, and server computer system 190 includes Web server 161. Web browser 121 interprets HTML (HyperText Mark-up Language) documents and other data retrieved by Web server 161.

In the present embodiment of the present invention, an executable program resides on server computer system 190, specifically on Web server 161. The executable program implements and manages the connection between server computer system 190, client computer system 120, and camera 300. The executable program can be implemented as a Java servlet, as a cgi-bin (Common Gateway Interface-binaries), or as a similar type of application.

With reference still to FIG. 1A, client computer system 120 is communicatively coupled to ISP 115 via a POTS (plain old telephone system) dial-up connection. Client computer system 120 is coupled to the Internet 150 via one of a bank of modems maintained on the premises of ISP 115. ISP 115 is coupled directly to the Internet via an all-digital connection (e.g., a T1 line). However, other means of coupling client computer system 120 to the Internet 150 may be used in accordance with the present invention.

As depicted in FIG. 1A, camera 300 is communicatively coupled to server computer system 190 via the Internet 150 using a dial-up connection to ISP 110 via a POTS line. Digital camera 300 accesses ISP 110 using a modem, coupling to one of a bank of modems maintained on the premises of ISP 110. ISP 110 is in turn coupled directly to the Internet 150 via an all-digital connection. However, other means of coupling camera 300 to the Internet 150 may be used in accordance with the present invention.

With reference still to FIG. 1A, it should be further appreciated that while communication network 100 shows camera 300 coupling to Internet 150 via one ISP (e.g., ISP 110) and user 120 coupling to Internet 150 via a separate ISP (e.g., ISP 115), user 120 and camera 300 could be coupled to Internet 150 through a single ISP. In such a case, user 120 and camera 300 would be coupled to two separate access ports (e.g., two separate modems out of a bank of modems) of the same ISP.

With reference now to FIG. 1B, in another embodiment of the present invention, the communication network is comprised of Local Area Network (LAN) 170. For example, LAN 170 may be a communication network located within a firewall of an organization or corporation. Client (or user) computer system 171 and server computer system 172 are communicatively coupled via communication line 173a. Client computer system 171 includes an application that is analogous to a Web browser for interpreting HTML documents and other data. Similarly, server computer system 172 includes an application analogous to a Web server for retrieving HTML documents and other data. Camera 300 can be coupled to server computer system 172 through any

8

of a variety of means known in the art. For example, camera 300 can be coupled to server computer system 172 via communication line 173b of LAN 170. This coupling can be accomplished over any network protocol that supports a persistent network connection, such as TCP (Transmission Control Protocol), NetBIOS, IPX (Internet Packet Exchange), and LU6.2, and link layers protocols such as Ethernet, token ring, and ATM (Asynchronous Transfer Mode). Alternatively, camera 300 can be coupled to server computer system 172 via an input/output port (e.g., a serial port) of server computer system 172.

Referring now to FIG. 1C, a more detailed diagram of camera 300 coupled to the Internet 150 is shown. Camera 300 is coupled to an external modem 101. Camera 300 is coupled to modem 101 via any of several communications means (e.g., Universal Serial Bus, infrared link, and the like). Modem 101 is in turn coupled to a POTS telephone jack 102 at the camera's location. Telephone jack 102 couples modem 101 to one of modems 103 of ISP 110 via the telephone company's local loop. ISP 110 is directly coupled to the Internet 150 via an all digital connection (e.g., a T1 line).

Continuing with reference to FIG. 1C, modem 101 is shown as an external modem. However, the functionality of modem 101 can be implemented directly within the electronics of camera 300 (e.g., via a modem application-specific integrated circuit or ASIC), or alternatively can be implemented as a software-only modem executing on a computer within camera 300. As such, it should be appreciated that, at the hardware connectivity level, modem 101 can take several forms. For example, a wireless modem can be used in which case the camera is not connected via an external wire to any land line. Alternatively, there may be applications in which camera 300 includes suitable electronic components enabling a connection to a conventional computer system network (e.g., Ethernet, AppleTalk, and the like), which is in turn directly connected to the Internet (e.g., via a gateway, a firewall, and the like), thereby doing away with the requirement for an ISP. Hence, it should be appreciated that the present invention is not limited to any particular method of accessing the Internet 150.

Refer now to FIG. 2 which illustrates server computer system 190 upon which embodiments of the present invention may be practiced (the following discussion is also pertinent to a client computer system). In general, server computer system 190 comprises bus 200 for communicating information, processor 201 coupled with bus 200 for processing information and instructions, random access memory 202 coupled with bus 200 for storing information and instructions for processor 201, read-only memory 203 coupled with bus 200 for storing static information and instructions for processor 201, data storage device 204 such as a magnetic or optical disk and disk drive coupled with bus 200 for storing information and instructions, optional display device 205 coupled to bus 200 for displaying information to the computer user, optional alphanumeric input device 206 including alphanumeric and function keys coupled to bus 200 for communicating information and command selections to processor 201, optional cursor control device 207 coupled to bus 200 for communicating user input information and command selections to processor 201, and network interface card (NIC) 208 coupled to bus 200 for communicating from a communication network to processor 201.

Display device 205 of FIG. 2 utilized with server computer system 190 of the present invention may be a liquid crystal device, cathode ray tube, or other display device

suitable for creating graphic images and alphanumeric characters recognizable to the user. Cursor control device 207 allows the computer user to dynamically signal the two-dimensional movement of a visible symbol (pointer) on a display screen of display device 205. Many implementations of the cursor control device are known in the art including a trackball, mouse, joystick or special keys on alphanumeric input device 206 capable of signaling movement of a given direction or manner of displacement. It is to be appreciated that the cursor control 207 also may be directed and/or activated via input from the keyboard using special keys and key sequence commands. Alternatively, the cursor may be directed and/or activated via input from a number of specially adapted cursor directing devices.

Referring now to FIG. 3, a block diagram of digital camera 300 is shown for use in accordance with the present invention. Camera 300 preferably comprises imaging device 314, system bus 316 and computer 318. Imaging device 314 is optically coupled to object 312 and electrically coupled via system bus 316 to computer 318. Once a photographer has focused imaging device 314 on object 312 and, using a capture button or some other means, instructed camera 300 to capture an image of object 312, computer 318 commands imaging device 314 via system bus 316 to capture raw data representing object 312. The captured raw data are transferred over system bus 316 to computer 318, which performs various processing functions on the data before storing it in memory. System bus 316 also passes various status and control signals between imaging device 314 and computer 318.

Referring now to FIG. 4, a block diagram of one embodiment of computer 318 is shown. System bus 316 provides connection paths between imaging device 314, an optional power manager 442, central processing unit (CPU) 444, dynamic random-access memory (DRAM) 446, input/output interface (I/O) 448, non-volatile memory 450, and buffers/connectors 452. Removable memory 454 connects to system bus 316 via buffers/connectors 452. Alternatively, camera 300 may be implemented without removable memory 454 or buffers/connectors 452.

Power manager 442 communicates via line 466 with power supply 456 and coordinates power management operations for camera 300. CPU 444 typically includes a conventional processor device for controlling the operation of camera 300. In the present embodiment, CPU 444 is capable of concurrently running multiple software routines to control the various processes of camera 300 within a multi-threaded environment. DRAM 446 is a contiguous block of dynamic memory which may be selectively allocated to various storage functions. LCD controller 490 accesses DRAM 446 and transfers processed image data to LCD screen 402 for display.

I/O 448 is an interface device allowing communications to and from computer 318. I/O 448 permits an external device (not shown) to connect to and communicate with computer 318. I/O 448 also interfaces with a plurality of buttons and/or dials 404, and optional status LCD 406, which in addition to LCD screen 402, are the hardware elements of the camera's user interface 408.

Non-volatile memory 450, which may typically comprise a conventional read-only memory or flash memory, stores a set of computer-readable program instructions to control the operation of camera 300.

Referring now to FIG. 5, a memory map showing one embodiment of dynamic random access memory (DRAM) 446 is shown. In the present embodiment, DRAM 446 includes RAM disk 532, system area 534, and working memory 530.

RAM disk 532 is a memory area used for storing raw and compressed data and typically is organized in a "sectored" format similar to that of conventional hard disk drives. In the present embodiment, RAM disk 532 uses a well-known and standardized file system to permit external devices, via I/O 448 of FIG. 4, to readily recognize and access the data stored on RAM disk 532. System area 534 typically stores data regarding system errors (for example, why a system shutdown occurred) for use by CPU 444 (FIG. 4) upon a restart of computer 318 (FIG. 3).

Working memory 530 includes various stacks, data structures and variables used by CPU 444 while executing the software routines used within computer 318. Working memory 530 also includes several input buffers 538 for temporarily storing sets of raw data received from imaging device 314 (FIG. 3), and frame buffer 536 for storing data for display on LCD screen 402 (FIG. 4). In the present embodiment, each input buffer 538 and frame buffer 536 are split into two separate buffers (shown by the dashed lines) to improve the display speed of the digital camera and to prevent the tearing of the image in LCD screen 402.

Referring now to FIG. 6, a diagram of the connectivity and application software of camera 300 of FIG. 3 is shown. At the software level, computer 318 (FIG. 3) of camera 300 hosts any network protocol that supports a persistent network connection. This coupling can be accomplished over any network protocol that supports a persistent network connection, such as TCP/IP (Transmission Control Protocol/Internet Protocol) including Point-to-Point Protocol, NetBIOS, IPX, and LU6.2, and link layers protocols such as Ethernet, token ring, and ATM. Protocol stack 601 interfaces with the communications hardware 602 (e.g., a modem) of camera 300 and the application layer 603. The bottom of protocol stack 601 includes communication hardware interface drivers which interface directly with the various communications hardware with which camera 300 must function (e.g., a Universal Serial Bus and the like). Applications layer 603, protocol stack 601, and operating system 604 are installed as software modules in DRAM 446 (FIG. 4) of camera 300. Software applications within applications layer 603 interface with operating system 604. Operating system 604 controls the hardware functionality of camera 300 (e.g., taking pictures, storing pictures, and the like) via camera hardware 605. Incoming data 620, such as HTTP (Hypertext Transfer Protocol) requests and the like, are received and outgoing data 630, such as HTML (Hypertext Markup Language) files and the like, are transferred to and from camera 300 via protocol stack 601 and communications hardware 602. Web browser 121 of FIG. 1A (or a program of similar function) can process data files, launch plug-ins, and run Java applets that communicate with camera 300 in a variety of methods in addition to those methods involving an exchange of HTML files.

FIG. 7 illustrates an executable program 700 for client computer system 120 (specifically, Web browser 121 of FIG. 1A or a program of similar function) to remotely access camera 300 (FIG. 3), where executable program 700 is implemented in accordance with the present invention as program instructions stored in computer-readable memory units (e.g., read-only memory 203) and implemented by processor 201 of server computer system 190 of FIG. 1A (specifically, by Web server 161). Executable program 700 performs functions both for and in response to Web browser 121 (or a program of similar function) and camera 300. The description below first discusses the steps associated with setting up a connection between executable program 700 and camera 300, then discusses the steps associated with

11

setting up a connection between executable program 700 and Web browser 121 (or a program of similar function), however, the present invention is not limited by the order in which these steps are presented.

In the present embodiment, executable program 700 is identified and accessed by its own unique address, commonly referred to as an URL (Unified Resource Locator), as is well known in the art. The URL for executable program 700 fully describes where it resides on a communication network (e.g., the Internet 150) and how it is accessed. In the present embodiment, included in the URL for executable program 700 is the name of camera 300. Accordingly, in one embodiment using a servlet for executable program 700, a standard format for a URL is: `http://webserver/HostName/cameraServlet/WellKnownName/cameraName`.

In step 705, executable program 700 receives and accepts a connection request from camera 300. Executable program 700 runs constantly on Web server 161 and is configured to listen for connection requests on a plurality of communication protocols (e.g., TCP, NetBIOS, and the like). In the present embodiment, camera 300 is connected to executable program 700 via Web server 161 as described in conjunction with FIG. 1A, and opens the connection using the connectivity and application software described in conjunction with FIG. 6.

In step 710, executable program 700 receives and reads registration information from camera 300. As mentioned above, executable program 700 is configured to communicate using a number of different communication protocols. Such registration information includes the name of the camera and authentication information such as security information and account information. Executable program 700 uses the camera name to identify the camera and locate it in response to a client request.

In step 720, executable program 700 compares the registration information with a predefined access control list to determine if camera 300 is a camera for which Web server 161 is to provide support and service. If not, executable program 700 closes the connection between Web server 161 and camera 300.

In step 725, upon successful completion of step 720, executable program 700 registers camera 300 and stores the camera's name and associated requirements, such as security and account information. Executable program 700 also sends a message that camera 300 is registered. At this point, the connection between executable program 700 can be either terminated or maintained at the option of the camera's operator. If the connection is terminated, the registration information is maintained by Web server 161 and can be later accessed by executable program 700 when a subsequent connection is made with camera 300.

In step 730, executable program 700 accepts the connection request from camera 300 and thus has a persistent and long term connection with camera 300. As described above, the connection can be an ongoing connection maintained from the time when camera 300 was first registered. New connections can be made in the future whenever camera 300 reinitiates the registration protocol. Once camera 300 and executable program 700 have established a connection, they then wait until a client also makes a connection to access the camera. However, as will be described below in conjunction with step 770, it is not necessary for there to be an open connection between the Web server and the camera at the same time that there is an open connection between the Web browser (or a program of similar function) and Web server to accomplish remote access of the camera in accordance with the present invention.

12

In step 740, executable program 700 receives and accepts a request for a connection from a client. The client enters the URL of executable program 700, including the name of the camera (e.g., camera 300) to which access is desired, into a Web browser (e.g., Web browser 121 on client computer system 120 of FIG. 1A or a program of similar function). Using standard communication protocols such as TCP/IP, Web server 161 is queried with the URL for executable program 700. Web server 161 recognizes the URL and makes the connection to executable program 700. Thus, the present invention establishes a single location identified by a known URL where the client always goes to connect to the camera, no matter where the camera is or where the client is. (For those cases in which TCP/IP is not available—for example, when the device is not attached to the Internet or the like—camera 300 connects to executable program 700 directly using a well-known address that is associated with the protocol being used, such as a NetBIOS name.)

In step 745, executable program 700 then assumes control over the TCP/IP connection between Web browser 121 (or a program of similar function) and Web server 161. Executable program 700 establishes a persistent and long term connection between the browser and server. That is, the connection between Web browser 121 (or a program of similar function) and Web server 161 is kept open by executable program 700. As will be described below in conjunction with step 770, it is not necessary for there to be an open connection between the client computer system and the Web server at the same time that there is an open connection between the Web server and the camera to accomplish remote access of the camera in accordance with the present invention.

In step 750, executable program 700 next determines if camera 300 is registered as discussed in conjunction with step 735. If camera 300 is not registered, executable program 700 sends an appropriate message to the client to indicate such.

In step 760, if the camera is registered, executable program 700 validates the required access information provided by the client against the security and account information provided when camera 300 was registered. For example, executable program 700 validates whether the client is utilizing an authorized password or user name. If not, executable program 700 transmits an appropriate message to the client. The present invention can optionally provide additional services related to security or account information. For example, it could control the type of access a client is permitted based on the authentication information received from the client, or it could verify credit information and bill the client for services requiring payment.

In step 765, upon satisfactory completion of step 760, executable program 700 extends the connection from Web browser 121 (or a program of similar function) to camera 300 if there is an established connection to the camera as described in conjunction with step 730. Hence, a client using Web browser 121 or a program of similar function has direct, remote access to camera 300 via executable program 700 in Web server 161.

In step 770, executable program 700 forwards commands from a client to camera 300, and forwards images and other data from camera 300 to the client via Web server 161 and Web browser 121 (or a program of similar function). That is, executable program 700 enables a direct communication between the client computer system and the camera allowing the client to remotely access and manage the camera. If the client and the camera are both concurrently connected to

13

executable program 700, then the client immediately receives the data, and camera 300 immediately executes any commands from the client.

However, if Web browser 121 (or a program of similar function) and camera 300 are not each connected at the same time, remote access to the camera is still accomplished in accordance with the present invention. If camera 300 is not on-line, a client uses Web browser 121 (or a program of similar function) to access Web server 161 and executable program 700. The client transmits commands, and executable program 700 stores the commands on Web server 161. The client may then close the connection to the Web server. Subsequently, when camera 300 opens a connection to Web server 161 and executable program 700, executable program 700 retrieves the commands and forwards them to the camera. Camera 300 downloads the commands and executes them. The results from executing the commands are then sent to executable program 700, which stores them on Web server 161 until they are retrieved by the client.

Similarly, if camera 300 establishes a connection with Web server 161 and executable program 700 but the client is not on-line, the camera can, for example, download images and any other data that executable program 700 stores on Web server 161. Camera 300 may then close the connection to the Web server. The client then later makes a connection to Web server 161 and executable program 700, which retrieves the data and forwards it to the client. The client can also enter commands at this time, which are stored by executable program 700 as described above.

In summary, with references to FIG. 1A and FIG. 7, the client, via Web browser 121 or a program of similar function, Web server 161 and executable program 700, accesses camera 300 to request and retrieve data. Web browser 121 or a program of similar function inserts the Internet address inside the data request (e.g., a HTTP request) and sends the request to Web server 161. Web server 161 receives the data request and associates the request with executable program 700, which in turn assumes the connection between Web server 161 and Web browser 121 (or a program of similar function), and which also establishes a connection between Web browser 121 (or a program of similar function) and camera 300. Executable program 700 subsequently forwards commands from the client to camera 300, and retrieves the requested data (e.g., a HTML file) containing the data and sends it back to Web browser 121 (or a program of similar function). Web browser 121 (or a program of similar function) then interprets the commands and displays the resulting image. The process of accessing a data file from a Web server is commonly referred to as accessing a Web page. Similarly, the process of sending data files from a Web server to a Web browser is commonly referred to as sending a Web page.

Thus, as described above, the present invention provides an intuitive and easy-to-use interface enabling remote access between a client and a camera. By functioning with widely used and familiar Web browsers (or programs of similar function) using standard format URLs to identify the executable program and camera, the present invention provides a simple and familiar interface for accessing the camera. By registering the camera and using an unchanging URL, name to identify the executable program and the camera, the present invention enables the client to locate and access the camera from any remote location no matter where the camera is located. In addition, by using a Java servlet or a cgi-bin for the executable program, the present invention is supported by commonly used Web servers and is readily implemented.

14

Executable program 700 is located on the Web server and not on camera 300, so it does not require additional and substantial memory dedicated to enabling remote access. As such, the present invention permits remote access within the constraints of the size of the camera. In addition, in accordance with the present invention, camera 300 does not require a separate, external computer system (e.g., a personal computer system) for connecting to ISP 110 (FIG. 1A) or for implementing commands and transmitting data, thus providing an inexpensive method for providing remote access to cameras.

Also, by locating the present invention on a Web server (e.g., Web server 161 of FIG. 1A), the Web server becomes a focal point for accessing and managing a plurality of cameras that otherwise would have to be managed and configured separately. For example, executable program 700 on Web server 161 could be updated with new software, and in effect all cameras accessed through that executable program would automatically be updated as well. As another example, executable program 700 could be configured to compile data regarding interactions between clients and all cameras accessed by the executable program. In another example, in accordance with the present invention, a client needs to go only to a single location to determine which of a plurality of cameras served by the executable program have data that have been downloaded to the Web server. Thus, instead of having to access a number of cameras separately, the present invention establishes a single location from which a client can access information about several cameras.

By functioning with a Web-based interface and widely used Web browsers (or programs of similar function), the present invention provides a simple, intuitive, and familiar interface for accessing camera 300's functionality. Accordingly, camera 300's controls and functions are intuitively easy to utilize. Since Web pages and their associated controls (e.g., push buttons, data entry fields, and the like) are very familiar to most users, the remote access functionality of camera 300 can be utilized without requiring a extensive learning period for new users. For example, a consumer purchasing a remotely accessible camera is typically able to easily and immediately use the remote accessibility functions with minimal set-up.

FIG. 8 illustrates a process 800 for remotely accessing camera 300 (FIG. 3), where process 800 is implemented as program instructions stored in computer-readable memory units (e.g., non-volatile memory 450 of FIG. 4) and implemented by CPU 444 (FIG. 4) of camera 300 in accordance with the present invention. FIG. 8 illustrates the process for remotely accessing a camera from the perspective of the actions performed by the camera and the camera operator in accordance with the present invention.

In step 805, camera 300 of FIG. 3 opens a connection to communication network 100 of FIG. 1A. In the present embodiment, camera 300 is coupled to server computer system 190 (specifically, Web server 161) as described in conjunction with FIG. 1A, and opens the connection using the connectivity and application software described in conjunction with FIG. 6. As described in conjunction with FIG. 1C, in the present embodiment, camera 300 couples directly to the telephone system such that a separate and dedicated computer system (e.g., a personal computer system) is not necessary.

In step 810, with the connection made to Web server 161, camera 300 requests a connection to executable program 700 (FIG. 7). For cases in which camera 300 is accessing

15

executable program 700 via, for example, a TCP/IP network such as the Internet 150, then executable program 700 is identified with a URL that is used by the camera to access the executable program. For those cases in which TCP/IP is not available (e.g., when the device is not attached to the Internet or the like), camera 300 connects to executable program 700 directly using a well-known address that is associated with the protocol being used, such as a NetBIOS name.

In step 820, with the camera connected to the executable program, camera 300 registers with executable program 700. For example, camera 300 provides information including an identification name and authentication information such as a password and account information. The information is electronically transmitted from camera 300 and read by executable program 700. Based on this information, the connection between camera 300 and executable program 700 is established if the camera is of the type that is designated to be supported by Web server 161.

In step 830, camera 300 and executable program 700 are linked via a persistent and long term connection; that is, the connection remains open awaiting a client to request access to the camera via Web server 161. As discussed above in conjunction with FIG. 7, it is not necessary for the camera and a client to be connected at the same time to executable program 700.

In step 835, camera 300 receives commands from and transmits data to a client using a Web browser or a program of similar function on a client computer system (e.g., client computer system 120 and Web browser 121 of FIG. 1A or a program of similar function). As described above in conjunction with FIG. 7, the commands and data can be transmitted through an active connection or stored on the Web server.

In the present embodiment, camera 300 is provided with several different operating modes for supporting various camera functions. In capture mode, camera 300 supports the actions of preparing to capture an image and of capturing an image. In review mode, camera 300 supports the actions of reviewing camera contents, editing and sorting images, and printing and transferring images. In play mode, camera 300 allows the client to view screen-sized images in the orientation that the image was captured. Play mode also allows the client to hear recorded sound associated to a displayed image, and to play back sequential groupings of images, which may comprise time lapse, slide show, and burst image images. The client preferably switches between the capture, review, and play modes.

Camera 300 is capable of implementing a wide variety of remote access and remote imaging/surveillance applications. In the present embodiment, camera 300 only records successive images for remote access by the client. The images are loaded into the camera's memory on a first-in, first-out (FIFO) basis, with the earliest recorded image being replaced by the latest recorded image. The number of images available to the client depends upon the amount of installed memory in the camera.

With reference still to step 835 of FIG. 8, when the client and the camera are connected to the Web server at the same time, the commands can be transmitted directly from Web browser 121 (or a program of similar function) to camera 300 via executable program 700 on Web server 161, and camera 300 then executes the commands on-line. Alternatively, when camera 300 is not connected to executable program 700, a client can store commands on Web server 161 by accessing the Web server in a normal fashion,

16

then entering the commands to be stored via executable program 700. Camera 300 then executes the commands when it subsequently connects with executable program 700.

Also in step 835, camera 300 transmits data to a client. Similar to the above, data from camera 300 can be transmitted directly to Web browser 121 or a program of similar function via executable program 700 on Web server 161, when the client and the camera are connected to the Web server at the same time. Alternatively, when a client is not connected to executable program 700, the data are stored on Web server 161 by executable program 700, which then retrieves and transmits the data to a client when the client subsequently connects with executable program 700.

This process of accessing camera 300 from Web browser 121 or a program of similar function occurs transparently with respect to the client. In a typical case, for example, the client types the URL for executable program 700 (which includes the name of camera 300) into Web browser 121 (or a program of similar function) and his "enter" or "return". In accordance with the present invention, the next Web page the client views is the image generated by the data returned from camera 300. Beyond entering the URL for executable program 700 including camera 300, no further action from the client is required in order to access the Web pages hosted by camera 300.

FIG. 9 illustrates a process 900 for remotely accessing camera 300 (FIG. 3), where process 900 is implemented as program instructions stored in computer-readable memory units (e.g., read-only memory) and implemented by the central processor of client computer system 120 (specifically, Web browser 121 or a program of similar function) of FIG. 1A. FIG. 9 illustrates the process for remotely accessing a camera from the perspective of the actions performed by the client in accordance with the present invention.

In step 905, the client opens a network connection such as a dial-up connection to ISP 115 of FIG. 1A.

In step 910, the client enters into a Web browser (e.g., Web browser 121 on client computer system 120 of FIG. 1A or a program of similar function) the URL of executable program 700, including the name of the camera (e.g., camera 300) to which access is desired. Alternatively, the client enters the URL of executable program 700 only. (For those cases in which TCP/IP is not available—for example, when the device is not attached to the Internet or the like—camera 300 connects to executable program 700 directly using a well-known address that is associated with the protocol being used, such as a NetBIOS name.) Using standard communication protocols such as TCP/IP, Web server 161 is queried with the URL for executable program 700. Web server 161 recognizes the URL and makes the connection to executable program 700. A persistent and long term two-way connection is opened between the client on Web browser 121 (or a program of similar function) and executable program 700.

In step 915, the client enters the authentication information required to gain access to executable program 700 and camera 300.

In step 920, as explained above, from his/her remote location the client sends commands to camera 300 and receives images from the camera. As explained above, the client and the camera do not have to be connected to the Web server at the same time to enable remote access. Depending on the particular application, the Web page for camera 300 can include control buttons, data entry fields, drop-down menus, and the like.

17

Thus, executable program 700 enables the client to access from a remote location the functional controls of camera 300 as well as the images and other data acquired by the camera.

Pseudo-Code Sections A, B, C, D and E below provide additional details regarding processes 700, 800 and 900 of FIGS. 7, 8 and 9. Pseudo-Codes Sections A through E represent the method of one embodiment of the present invention. However, it is appreciated that other embodiments are possible in accordance with the present invention.

18

With references to FIGS. 7 and 9, the pseudo-code for the connection of a client to the server computer system, specifically to executable program 700, is described in accordance with one embodiment of the present invention in Pseudo-Code Section A below.

Pseudo-Code Section A: Client Connection Setup

```

1. Client sends HTTP Post command to http://serverName/gateway device="deviceName"
   Note: authentication/security data is optional and is handled by the browser.
2. Client receives HTTP response from Gateway
3. If response is OK or OKDataPresent
{
    Client has TCP connection it can use to send/receive anything to/from the device
    The protocol used can be any that the Client & Device agree upon.
}
else
{
    The response is failed or DataPresent.
    The TCP connection is closed:
}

```

Note: the OKDataPresent and DataPresent responses are informational. The Client at any time on separate connections may issue a CacheData or GetCache command to the Gateway. See algorithms C.

30 With references to FIGS. 7 and 8, the pseudo-code for the connection of a device (e.g., digital camera 300 of FIG. 1A) to the server computer system, specifically to executable program 700, is described in accordance with one embodiment of the present invention in Pseudo-Code Section B below.

Pseudo-Code Section B: Device Connection Setup/Registration

```

1. Device establishes a connection to the Gateway using any networking protocol
   supported by the Gateway (TCP, NetBIOS, IPX, 802.2, etc) using a known address.
2. Device sends a Register request to the Gateway on the connection passing
   its name (and optionally authentication/security information).
3. Device receives response from Gateway
4. If the response is OK or OKDataPresent
{
    The Device may use the existing connection to wait for requests from clients
    or the Device may optionally close the connection.
}
else
{
    The response is Failed or DataPresent
    The connection is closed.
}

```

Note: the OKDataPresent and DataPresent responses are informational. The Device at any time on separate connections may issue a CacheData or GetCache command to the Gateway. See algorithm D.

60

With reference to FIG. 7, the pseudo-code for executable program 700 in response to a client connection request is described in accordance with one embodiment of the present invention in Pseudo-Code Section C below.

65 Pseudo-Code Section C: Gateway Handles Client Connection Request

```

Start:
    Wait for incoming requests from clients
    if its an HTTP Post command with parameters CacheData and deviceName
    {
        go to CacheData
    }
    else if its an HTTP Get command with parameters GetCache and deviceName
    {
        go to GetCache
    }
    if its an HTTP Post command from a client with the name of device
    {
        go to Connect
    }
    else
    {
        go to Fail
    }
Connect:
    Look up device by name in registry
    if the device entry is found
    {
        if there is a device connection id in the entry
        {
            if the entry is not busy
                go to Success
            else go to Fail
        }
        else
        {
            Attempt to establish connection with device at last known address
            if connection established
            {
                go to Success
            }
            else go to Fail
        }
    }
    else
    {
        send Fail response
        close connection
        go to Start
    }
}
Fail:
    if there is data in the cache
    {
        set response to DataPresent
    }
    else
    {
        set response to Fail
    }
    send response
    close connection
    go to Start
Success:
    if there is data in the cache
    {
        set response to OkDataPresent
    }
    else
    {
        set response to Ok
    }
    store connection id of the client in the registry entry
    go to Start
CacheData:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        store data in Cache for device with identification information
    }

```

-continued

```

    send Ok response
    close connection
    go to Start;
}
GetCache:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        send data in Cache for this client
        close connection
        go to Start
    }
}

```

With reference to FIG. 7, the pseudo-code for executable program 700 in response to a device connection request is described in accordance with one embodiment of the present invention in Pseudo-Code Section D below.

Pseudo-Code Section D: Gateway Handles Device Requests

```

Start:
    Wait for incoming requests from devices
    Receive incoming request
    if its a Register request
    {
        go to Register
    }
    else if its a CacheData request
    {
        go to CacheData
    }
    else if its a GetCache request:
    {
        go to GetCache
    }
    else
    {
        return Failed
        close connection
        go to Start
    }
Register:
    Note: The gateway may optionally authenticate device
    Get device name from Register request
    look up name in table
    if name is found
    {
        go to Success
    }
    else
    {
        The gateway may optionally add the name or reject the
        registration attempt, then go to Success or Fail
        respectively
    }
Success:
    check for the presence of cached client data
    if cached data is present
    {
        set response to OkDataPresent;
    }
    else
    {
        set response to Ok
    }
}

```

-continued

```

    Store a connection id for the incoming connection in
    the registry entry for the device.
    send response
    go to Start
Fail:
    if cached data from clients is present for this device
    {
        set response to DataPresent
    }
    else
    {
        set response to Fail
    }
    send response
    close connection
    go to Start
CacheData:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        store data in Cache for client
        send Ok response
        close connection
        go to Start;
    }
GetCache:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        send data in Cache for device to the device
        close connection
        go to Start
    }
}

```

With reference to FIG. 7, the pseudo-code for executable program 700, specifically the handling of data when an open connection is present between the device and the client

computer system, is described in accordance with one embodiment of the present invention in Pseudo-Code Section E below.

Pseudo-Code Section E: Gateway Handles Data on Existing Connections

```

Start:
  Wait for data
  Map the incoming connection id to its partner connection id
  if there is incoming data
  {
    Send the data out on the partner connection id
    go to Start
  }
  else if the connection closed
  {
    if it is a client connection
    {
      Remove the client's connection id from the
      device's entry in the registry.
    }
    else
    {
      Remove both the client and device connection ids
      from the device's entry in the registry.
      Close the client's connection
      go to Start
    }
  }
  go to Start
  
```

As described above, the remote accessibility of camera **300** provides for many new applications of digital imagery. One such application involves setting up camera **300** at some remote location and using it to take pictures at successive intervals. These pictures would be accessed via the Internet **150** as they are taken. The interval can be adjusted (e.g., more or less pictures per minute) in response to commands entered by a client via a Web browser (e.g., Web browser **121** of FIG. 1A or a program of similar function).

Another application involves using camera **300** in conjunction with a motion detector. When used in conjunction with a motion detector, camera **300** can be configured to capture an image in response to receiving a signal from the motion detector (e.g., detecting the motion of an intruder), thereby taking a picture of whatever triggered the detector's signal output. Alternatively, camera **300** can detect motion by simply comparing successive images to detect changes between them, thereby dispensing with the need for a separate motion detector.

Yet another application involves using camera **300** in conjunction with a remote aiming device. Camera **300** can be mounted on a remotely operated aiming device (e.g., a motorized tripod). The aiming device is controlled via the Internet **150** in the same manner the camera is controlled via the Internet **150**. Alternatively, camera **300** could be coupled to control the remote aiming device directly. The remote aiming device allows a client to control the field of view of the camera **300** in the same manner the client controls other functionality (e.g., picture resolution, picture interval, and the like).

In this manner, executable program **700** of the present invention is able to implement sophisticated remote surveillance of the type previously performed by expensive, prior art closed circuit television devices. Unlike the prior art, however, executable program **700** is inexpensive and relatively simple to implement.

Thus, the present invention provides a method for making a digital camera and its internally stored data remotely

accessible. The present invention enables the digital camera to be set to continuously take pictures of scenes and items of interest and to allow a user to access those pictures at any time. The present invention implements remote accessibility via a communication network such as the Internet, thus allowing the user to access the digital camera from virtually an unlimited number of locations and with the camera in virtually any location.

A digital camera in accordance with the present invention does not require a separate, external computer system (e.g., a personal computer system) for Internet connectivity, thus providing an inexpensive method for making remotely accessible digital cameras widely available. In addition, a digital camera in accordance with the present invention is accessed via the widely used and very familiar Web browser (or a program of similar function). By functioning with typical, widely used Web browsers (or programs of similar function), the present invention provides a simple, intuitive, and familiar interface for remotely accessing the digital camera's functionality and capabilities. In so doing, the controls and functions of the digital camera are intuitively easy to utilize, and do not require an extensive learning period for new users. The present invention also provides an efficient and user-transparent process of obtaining the address of a digital camera. Also, the present invention provides a method for efficiently administering a plurality of separate digital cameras.

Although the present invention is described in the context of a digital camera, it is not limited to this embodiment. Hence, the present invention does not provide only a limited degree of functionality as in the prior art applications; that is, it is not limited to either chat or video conferencing, or the like. As such, the present invention is capable of establishing a connection between any type of client system and remote device.

The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.

What is claimed is:

1. A method for allowing a client computer to remotely access a digital image capture unit via a communication network wherein a server computer system and said client computer system are communicatively coupled with communication equipment, said method comprising the steps of:
 - a) allowing said digital image capture unit to establish communication with said server computer system over said network, wherein said digital image capture unit includes connectivity software that enables said digital image capture unit to establish a network connection with said server computer system;
 - b) receiving an address of said digital image capture unit and registering said address in an executable program on said server computer system, wherein said digital image capture unit automatically registers said address with said server computer system;
 - c) allowing said client computer system to access said executable program on said server computer system; and

25

- d) establishing direct communication between said client computer system and said digital image capture unit by communicating commands between said client computer system and said digital image capture unit via said executable program, such that data captured by said digital image capture unit is transferred to said client computer system via said server computer system.
2. The method of claim 1 wherein said communication network is the Internet.
3. The method of claim 1 wherein said communication network is a Local Area Network.
4. The method of claim 1 wherein said image capture unit is a digital camera.
5. The method of claim 1 wherein step a) further comprises communicating authentication information between said digital image capture unit and said executable program.
6. The method of claim 1 wherein said executable program is a Java servlet.
7. The method of claim 1 wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).
8. The method of claim 1 wherein step c) further comprises connecting said digital image capture unit to said server computer system via a Local Area Network.
9. The method of claim 1 wherein step c) further comprises connecting said digital image capture unit to said server computer system via an input/output port of said server computer system.
10. The method of claim 1 wherein step c) further comprises connecting said digital image capture unit to said server computer system via the Internet.
11. The method of claim 1 wherein step d) further comprises storing said commands in a memory unit of said server computer system and communicating said commands to said digital image capture unit at a time when a connection is made between said server computer system and said digital image capture unit.
12. The method of claim 1 further comprising the steps of:
- e) accessing via said executable program data acquired by said digital image capture unit; and
 - f) transferring said data from said digital image capture unit to said client computer system via said server computer system.
13. The method of claim 12 further comprising storing said data in a memory unit of said server computer system and communicating said data to said client computer system at a time when a connection is made between said server computer system and said client computer system.
14. A computer system comprising:
- a) a processor coupled to a bus; and
 - a memory unit coupled to said bus and having stored therein an executable program that when executed by said processor implements a method for allowing a client computer to remotely access a digital image capture unit via a communication network, said method comprising the steps of:
 - a) allowing said digital image capture unit to establish communication with said server computer system over said network, wherein said digital image capture unit includes connectivity software that enables said digital image capture unit to establish a network connection with said server computer system;
 - b) receiving an address of said digital image capture unit and registering said address with said executable program, wherein said digital image capture unit automatically registers said address with said server computer system;

26

- c) allowing said client computer system to access said executable program over said communication network; and
 - d) establishing direct communication between said client computer system and said digital image capture unit by communicating commands between said client computer system and said digital capture unit via said executable program.
15. The computer system of claim 14 wherein said computer system is a server computer system.
16. The computer system of claim 14 wherein said digital image capture unit is a digital camera.
17. The computer system of claim 14 wherein said executable program is a Java servlet.
18. The computer system of claim 14 wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).
19. The computer system of claim 14 wherein said digital image capture unit and said computer system are connected via a Local Area Network.
20. The computer system of claim 14 wherein said digital image capture unit and said computer system are connected via an input/output port of said computer system.
21. The computer system of claim 14 wherein said digital image capture unit and said computer system are connected via the Internet.
22. In a communication network comprising a server computer system and a client computer system communicatively coupled with communication equipment, a method for allowing said client computer to remotely access a digital camera via said communication network, said method comprising the steps of:
- a) allowing said digital camera to establish communication with said server computer system over said communication network, wherein said digital camera includes connectivity software that enables said digital camera to establish a network connection with said server computer system;
 - b) receiving an address of said digital camera and registering said address in an executable program on said server computer system, wherein said digital camera automatically registers said address with said server computer system;
 - c) allowing said client computer system to access said executable program on said server computer system;
 - d) establishing direct communication between said client computer system and said digital camera by communicating commands between said client computer system and said digital camera via said executable program;
 - e) accessing via said server computer system data acquired by said digital camera; and
 - f) transferring said data from said digital camera to said client computer system via said server computer system.
23. The method of claim 22 wherein said executable program is a Java servlet.
24. The method of claim 22 wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).
25. A method for allowing a client computer to remotely access a digital image capture device via a communication network wherein a server computer system and said client computer system are communicatively coupled with communication equipment, said method comprising the steps of:
- a) allowing said digital image capture device to establish communication with said server computer system over

27

said communication network, wherein said digital image capture device includes connectivity software that enables said digital image capture device to establish a network connection with said server computer system;

- b) receiving an address of said digital image capture device and registering said address in an executable program on said server computer system, wherein said digital image capture device automatically registers said address with said server computer system;
- c) allowing said client computer system to access said executable program on said server computer system; and

28

- d) establishing direct communication between said client computer system and said digital image capture device by communicating commands between said client computer system and said digital image capture device via said executable program, such that data from said digital image capture device is transferred to said client computer system via said server computer system.

26. The method of claim **25** wherein said executable program is a Java servlet.

27. The method of claim **25** wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).

* * * * *



US00662218B2

(12) **United States Patent**
Mighdoll et al.

(10) Patent No.: **US 6,662,218 B2**
(45) Date of Patent: ***Dec. 9, 2003**

(54) **METHOD OF TRANSCODING DOCUMENTS
IN A NETWORK ENVIRONMENT USING A
PROXY SERVER**

(75) Inventors: **Lee S. Mighdoll**, San Francisco, CA
(US); **Bruce A. Leak**, Palo Alto, CA
(US); **Stephen C. Perlman**, Mountain
View, CA (US); **Phillip Y. Goldman**,
Los Altos, CA (US)

(73) Assignee: **WebTV Networks, Inc.**, Mountain
View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 0 days.

This patent is subject to a terminal dis-
claimer.

(21) Appl. No.: **10/247,885**

(22) Filed: **Sep. 20, 2002**

(65) **Priority Publication Data**

US 2003/0014499 A1 Jan. 16, 2003

Related U.S. Application Data

(63) Continuation of application No. 09/343,067, filed on Jun.
29, 1999, now abandoned, which is a continuation of application
No. 08/656,924, filed on Jun. 3, 1996, now Pat. No.
5,918,013.

(51) **Int. Cl.** **G06F 15/16**

(52) **U.S. Cl.** **709/219; 709/203; 709/217;
709/228; 709/229; 709/246**

(58) **Field of Search** **709/200-203,
709/217-219, 227-229, 231-232, 246-247**

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,575,579 A 3/1986 Simon et al. 178/4
4,852,151 A 7/1989 Dittakavi et al. 379/97

4,922,523 A 5/1990 Hashimoto 379/96
4,975,944 A 12/1990 Cho 379/209
4,995,074 A 2/1991 Goldnan et al. 379/97
5,005,011 A 4/1991 Perlman et al. 340/728
5,095,494 A 3/1992 Takahashi et al. 375/10
5,220,420 A 6/1993 Hoarty et al. 358/86
5,241,587 A 8/1993 Horton et al. 379/92
5,263,084 A 11/1993 Chaput et al. 379/215
5,287,401 A 2/1994 Lin 379/98
5,299,307 A 3/1994 Young 395/161

(List continued on next page.)

OTHER PUBLICATIONS

Rosoff, Matt, Review: "Gateway Destination PC," c/net
inc., 2 pages, Feb. 19, 1996.

Seidman, Robert, Article: "What Larry and Lou Know (That
You Don't)," c/net inc., 2 pages, Jan. 29, 1996.

Stellin, Susan, Article: "The \$500 Web Box: Less is More?"
c/net inc., 2 pages, 1996.

(List continued on next page.)

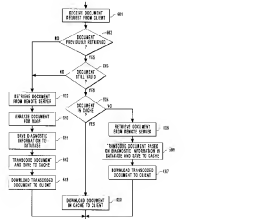
Primary Examiner—Bharat Barot

(74) Attorney, Agent, or Firm—Workman, Nydegger

(57) **ABSTRACT**

A method is described of providing a document to a client
coupled to a server. The server functions as a proxy on
behalf of the client for purposes of accessing a remote
server. In the method, a document is retrieved from the
remote server in response to a request from the client. The
document includes data to be used by the client in generating
a display. The proxying server alters (i.e., transcodes) the
data in the document to form a transcoded document. The
transcoded document is then transmitted to the client. The
proxying server transcodes the data in the document in order
to perform at least one of the following functions: (1)
matching decompression requirements at the client; (2)
converting the document into a format compatible for the
client; (3) reducing latency experienced by the client; and
(4) altering the document to fit into smaller memory space.

31 Claims, 12 Drawing Sheets



U.S. PATENT DOCUMENTS

5,325,423 A	6/1994	Lewis	379/60	5,678,041 A	10/1997	Baker et al.	395/609
5,329,619 A	7/1994	Page et al.	395/200	5,727,159 A	3/1998	Kikinis	395/200.76
5,341,293 A	8/1994	Verletney et al.	364/419.17	5,842,216 A	11/1998	Anderson et al.	707/203
5,369,688 A	11/1994	Tsukamoto et al.	379/100	5,889,955 A	3/1999	Shinozaki et al.	395/200.54
5,469,540 A	11/1995	Powers, III et al.	395/158	5,918,013 A	* 6/1999	Mighdoli et al.	709/217
5,488,411 A	1/1996	Fewis	348/8	5,978,817 A	11/1999	Gianandrea et al.	707/501
5,490,208 A	2/1996	Remillard	379/96	5,996,022 A	* 11/1999	Krueger et al.	709/247
5,530,852 A	6/1996	Meske, Jr. et al.	395/600	6,018,619 A	1/2000	Allard et al.	395/200.54
5,538,255 A	7/1996	Barker	463/41	6,226,412 B1	5/2001	Schwab	382/232
5,558,339 A	9/1996	Periman	463/42	OTHER PUBLICATIONS			
5,561,709 A	10/1996	Remillard	379/96	<i>Administrator's Guide, Netscape Proxy Server Version 2.0,</i>			
5,564,001 A	10/1996	Lewis	395/154	Netscape Communications Corporation, pp. 19–20, 1996.			
5,572,643 A	11/1996	Judson	395/793	Chankhuthod, Anawat, et al., "A Hierarchical Internet			
5,586,257 A	12/1996	Periman	463/42	<i>Object Cache,</i> " 1996 USEWIX Technical Conference (6			
5,586,260 A	12/1996	Hu	395/200.2	pages).			
5,612,730 A	3/1997	Lewis	348/8	Farrow, Rik, "Securing the Web: fire walls, proxy servers,			
5,623,600 A	4/1997	Ji et al.	395/187.01	<i>and data driven attacks,</i> " InfoWorld, Jun. 19, 1995, vol. 7,			
5,654,886 A	8/1997	Zereski, Jr. et al.	364/420	No. 25, pp. 103–104.			
5,657,390 A	8/1997	Elganai et al.	380/49	* cited by examiner			
5,657,450 A	8/1997	Rao et al.	395/610				

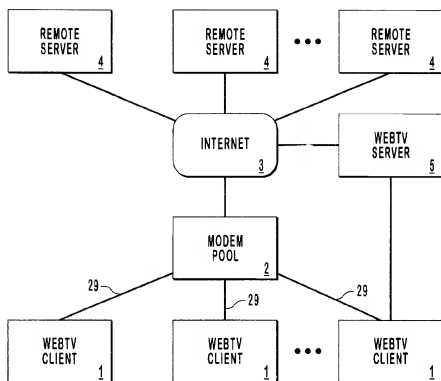


FIG. 1

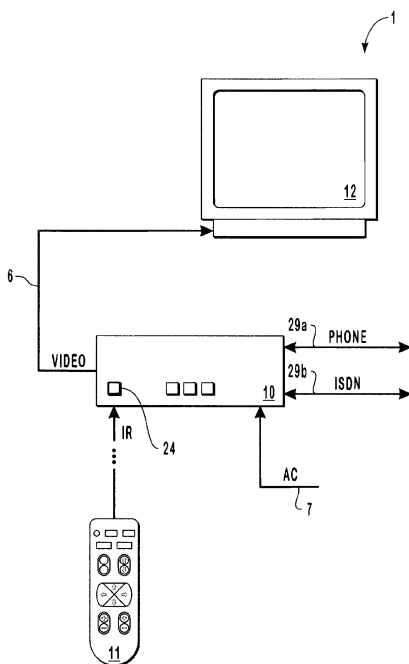


FIG. 2

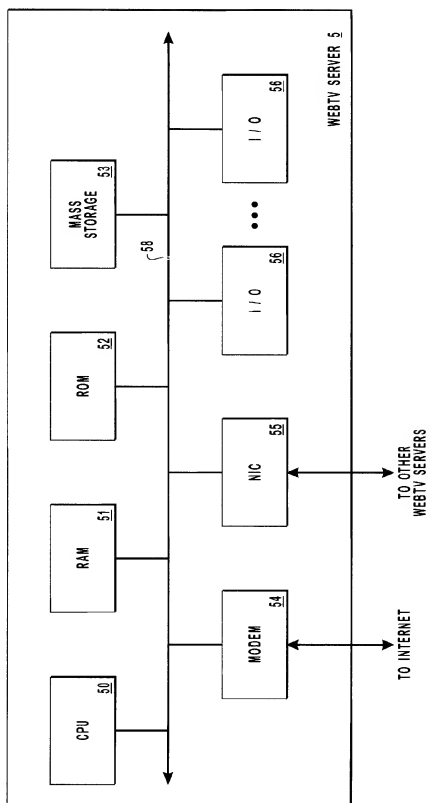


FIG. 3

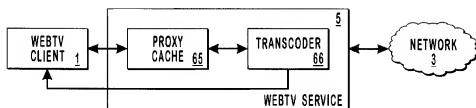


FIG. 4A

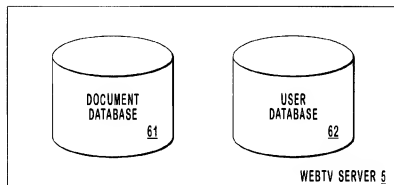


FIG. 4B

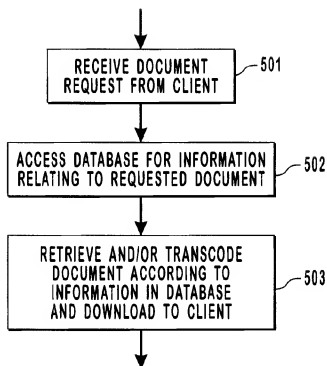


FIG. 5

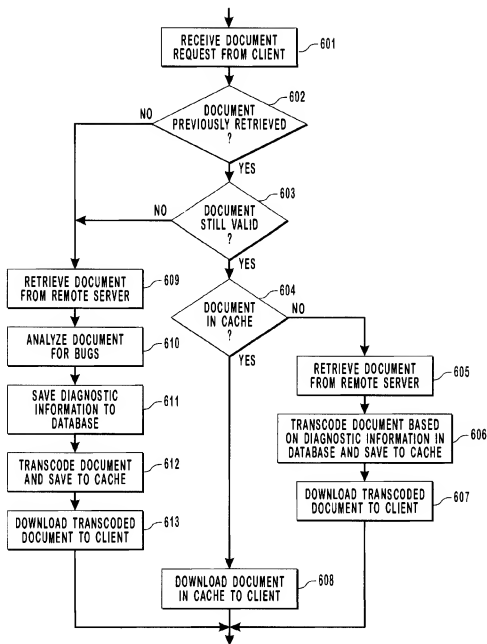


FIG. 6

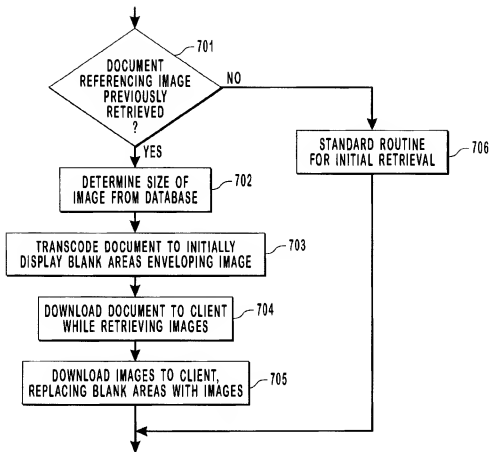


FIG. 7

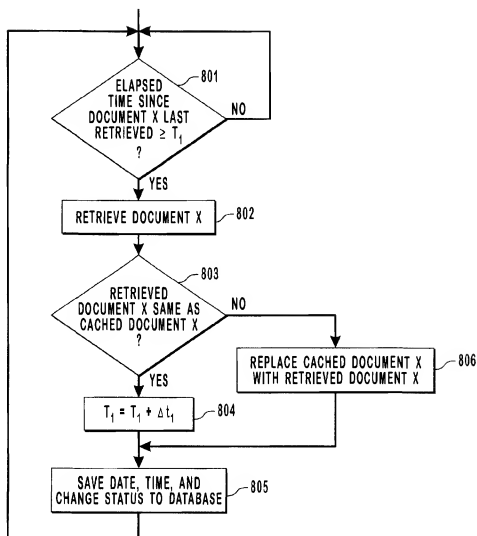


FIG. 8

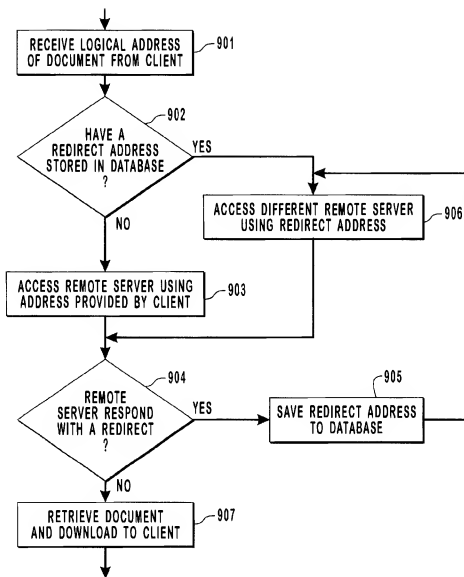


FIG. 9

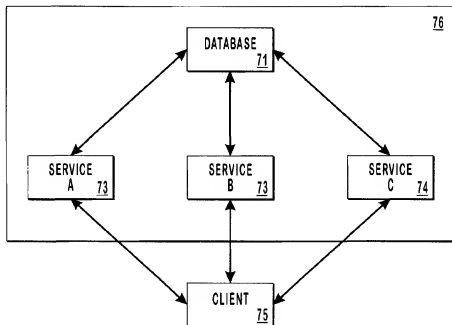


FIG. 10
(PRIOR ART)

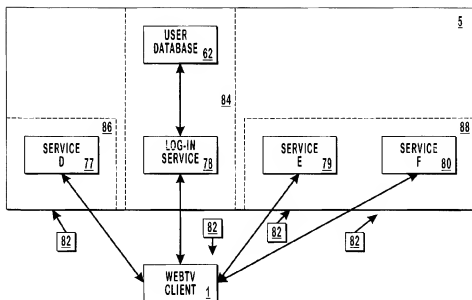


FIG. 11

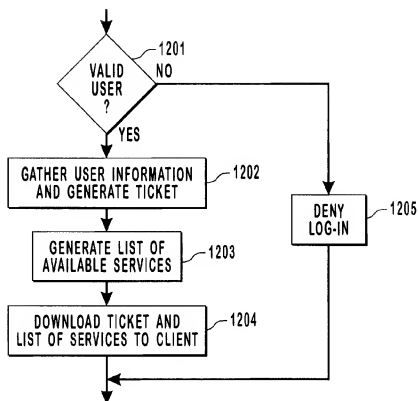


FIG. 12

1

METHOD OF TRANSCODING DOCUMENTS IN A NETWORK ENVIRONMENT USING A PROXY SERVER

RELATED APPLICATION

This is a continuation of U.S. patent application Ser. No. 09/343,067, entitled "Method of Transcoding Documents in a Network Environment Using a Proxy Server," filed Jun. 29, 1999, now abandoned which is a continuation of U.S. application Ser. No. 08/656,924 entitled "Method of Transcoding Documents in a Network Environment Using a Proxy Server," filed Jun. 3, 1996, now U.S. Pat. No. 5,918, 013 both of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. The Field of the Invention

The present invention pertains to the field of client-server computer networking. More particularly, the present invention relates to a method of transcoding documents in a network environment using a proxy server.

2. The Prior State of the Art

The number of people using personal computers has increased substantially in recent years, and along with this increase has come an explosion in the use of the Internet. One particular aspect of the Internet which has gained widespread use is the World-Wide Web ("the Web"). The Web is a collection of formatted hypertext pages located on numerous computers around the world that are logically connected by the Internet. Advances in network technology and software providing user interfaces to the Web ("Web browsers") have made the Web accessible to a large segment of the population. However, despite the growth in the development and use of the Web, many people are still unable to take advantage of this important resource.

Access to the Web has been limited thus far mostly to people who have access to a personal computer. However, many people cannot afford the cost of even a relatively inexpensive personal computer, while others are either unable or unwilling to learn the basic computer skills that are required to access the Web. Furthermore, Web browsers in the prior art generally do not provide the degree of user-friendliness desired by some people, and many computer novices do not have the patience to learn how to use the software. Therefore, it would be desirable to provide an inexpensive means by which a person can access the Web without the use of a personal computer. In particular, it would be desirable for a person to be able to access the Web pages using an ordinary television set and a remote control, so that the person feels more as if he or she is simply changing television channels, rather than utilizing a complex computer network.

Prior art Web technology also has other significant limitations which can make a person's experience unpleasant when browsing the Web. Web documents are commonly written in HTML (Hypertext Mark-up Language). HTML documents sometimes contain bugs (errors) or have features that are not recognized by certain Web browsers. These bugs or quirks in a document can cause a Web browser to fail. Thus, what is needed is a means for reducing the frequency with which client systems fail due to bugs or quirks in HTML documents.

Another problem associated with browsing the Web is latency. People commonly experience long, frustrating delays when browsing the Web. It is not unusual for a person to have to wait minutes after selecting a hypertext link for a

2

Web page to be completely downloaded to his computer and displayed on his computer screen. There are many possible causes for latency, such as heavy communications traffic on the Internet and slow response of remote servers. Latency can also be caused by Web pages including images. One reason for this effect is that, when an HTML document references an image, it takes time to retrieve the image itself after the referencing document has been retrieved. Another reason is that, in the prior art, if the referencing document does not specify the size of the image, the client system generally cannot display the Web page until the image itself has been retrieved. Numerous other sources of latency exist with respect to the Web. Therefore, what is needed is a means for reducing such latency, to eliminate some of the frustration which typically has been associated with browsing the Web.

Security is another concern associated with the Internet. Internet service providers (ISPs) generally maintain certain information about each customer in a database. This information may include information which a customer may not wish to become publicly known, such as social security numbers and credit card numbers. Maintaining the confidentiality of this information in a system that is connected to an expensive publicly-accessible computer network like the Internet can be problematic. Further, the problem can be aggravated by the fact that an ISP often provides numerous different services, each of which has access to this database. Allowing access to the database by many different entities creates many opportunities for security breaches to occur. Therefore, what is needed is a way to improve the security of confidential customer information in a server system coupled to the Internet.

SUMMARY AND OBJECTS OF THE INVENTION

A method is described of providing a document to a client coupled to a server. The server functions as a proxy on behalf of the client for purposes of accessing a remote server. In the method, a document is retrieved from the remote server in response to a request from the client. The document includes data to be used by the client in generating a display. The proxying server alters (i.e., transcodes) the data in the document to form a transcoded document. The transcoded document is then transmitted to the client.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follows. The proxying server transcodes the data in the document in order to perform at least one of the following functions: (1) matching decompression requirements at the client; (2) converting the document into a format compatible for the client; (3) reducing latency experienced by the client; and (4) altering the document to fit into smaller memory space.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follow.

BRIEF DESCRIPTION OF THE DRAWINGS

In order that the manner in which the above-recited and other advantages and objects of the invention are obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered limiting of its scope, the invention will be

3

described and explained with additional specificity and detail through the use of the accompanying drawings in which:

FIG. 1 illustrates several clients connected to a proxying server in a network;

FIG. 2 illustrates a client according to the present invention;

FIG. 3 is a block diagram of a server according to the present invention;

FIG. 4A illustrates a server including a proxy cache and a transcoder;

FIG. 4B illustrates databases used in a server according to the present invention;

FIG. 5 is a flow diagram illustrating a routine for transcoding a document retrieved from a remote server using data stored in a persistent database;

FIG. 6 is a flow diagram illustrating a routine for transcoding an HTML document for purposes of eliminating bugs or undesirable features;

FIG. 7 is a flow diagram illustrating a routine for reducing latency when downloading a document referencing an image to a client;

FIG. 8 is a flow diagram illustrating a routine for updating documents stored in the proxy cache using data stored in a persistent database;

FIG. 9 is a flow diagram illustrating a routine used by a server for retrieving documents from another remote server;

FIG. 10 is a block diagram of a prior art server system showing a relationship between various services and a database;

FIG. 11 is a block diagram of a server system according to the present invention showing a relationship between various services and a user database; and

FIG. 12 is a flow diagram illustrating a routine used by a server for regulating access to various services provided by the server.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

A method and apparatus are described for providing proxying and transcoding of documents in a network. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be evident, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

The present invention includes various steps, which will be described below. The steps can be embodied in machine-executable instructions, which can be used to cause a general-purpose or special-purpose processor programmed with the instructions to perform the steps. Alternatively, the steps of the present invention might be performed by specific hardware components that contain hardwired logic for performing the steps, or by any combination of programmed computer components and custom hardware components.

I. System Overview

The present invention is included in a system, known as WebTV™, for providing a user with access to the Internet. A user of a WebTV™ client generally accesses a WebTV™ server via a direct-dial telephone (POTS, for "plain old

4

telephone service"), ISDN (Integrated Services Digital Network), or other similar connection, in order to browse the Web, send and receive electronic mail (e-mail), and use various other WebTV™ network services. The WebTV™ network services are provided by WebTV™ servers using software residing within the WebTV™ servers in conjunction with software residing within a WebTV™ client.

FIG. 1 illustrates a basic configuration of the WebTV™ network according to one embodiment. A number of WebTV™ clients 1 are coupled to a modem pool 2 via direct-dial, bi-directional data connections 29, which may be telephone (POTS, i.e., "plain old telephone service"), ISDN (Integrated Services Digital Network), or any other similar type of connection. The modem pool 2 is coupled typically through a router, such as that conventionally known in the art, to a number of remote servers 4 via a conventional network infrastructure 3, such as the Internet. The WebTV™ system also includes a WebTV™ server 5, which specifically supports the WebTV™ clients 1. The WebTV™ clients 1 each have a connection to the WebTV™ server 5 either directly or through the modem pool 2 and the Internet 3. Note that the modem pool 2 is a conventional modem pool, such as those found today throughout the world providing access to the Internet and private networks.

Note that in this description, in order to facilitate explanation the WebTV™ server 5 is generally discussed as if it were a single device, and functions provided by the WebTV™ services are generally discussed as being performed by such single device. However, the WebTV™ server 5 may actually comprise multiple physical and logical devices connected in a distributed architecture, and the various functions discussed below which are provided by the WebTV™ services may actually be distributed among multiple WebTV™ server devices.

II. Client System

FIG. 2 illustrates a WebTV™ client 1. The WebTV™ client 1 includes an electronics unit 10 (hereinafter referred to as "the WebTV™ box 10"), an ordinary television set 12, and a remote control 11. In an alternative embodiment of the present invention, the WebTV™ box 10 is built into the television set 12 as an integral unit. The WebTV™ box 10 includes hardware and software for providing the user with a graphical user interface, by which the user can access the WebTV™ network services, browse the Web, send e-mail, and otherwise access the Internet.

The WebTV™ client 1 uses the television set 12 as a display device. The WebTV™ box 10 is coupled to the television set 12 by a video link 6. The video link 6 is an RF (radio frequency), S-video, composite video, or other equivalent form of video link. In the preferred embodiment, the client 1 includes both a standard modem and an ISDN modem, such that the communication link 29 between the WebTV™ box 10 and the server 5 can be either a telephone (POTS) connection 29a or an ISDN connection 29b. The WebTV™ box 10 receives power through a power line 7.

Remote control 11 is operated by the user in order to control the WebTV™ client 1 in browsing the Web, sending e-mail, and performing other Internet-related functions. The WebTV™ box 10 receives commands from remote control 11 via an infrared (IR) communication link. In alternative embodiments, the link between the remote control 11 and the WebTV™ box 10 may be RF or any equivalent mode of transmission.

III. Server System

The WebTV™ server 5 generally includes one or more computer systems generally having the architecture illus-

5

trated in FIG. 3. It should be noted that the illustrated architecture is only exemplary, the present invention is not constrained to this particular architecture. The illustrated architecture includes a central processing unit (CPU) 50, random access memory (RAM) 51, read-only memory (ROM) 52, a mass storage device 53, a modem 54, a network interface card (NIC) 55, and various other input/output (I/O) devices 56. Mass storage device 53 includes a magnetic, optical, or other equivalent storage medium. I/O devices 56 may include any or all of devices such as a display monitor, keyboard, cursor control device, etc. Modem 54 is used to communicate data to and from remote servers 4 via the Internet.

As noted above, the WebTV™ server 5 may actually comprise multiple physical and logical devices connected in a distributed architecture. Accordingly, NIC 55 is used to provide data communication with other devices that are part of the WebTV™ services. Modem 54 may also be used to communicate with other devices that are part of the WebTV™ services and which are not located in close geographic proximity to the illustrated device.

According to the present invention, the WebTV™ server 5 acts as a proxy in providing the WebTV™ client 1 with access to the Web and other WebTV™ services. More specifically, WebTV™ server 5 functions as a "caching proxy." FIG. 4A illustrates the caching feature of the WebTV™ server 5. In FIG. 4A, the WebTV™ server 5 is functionally located between the WebTV™ client 1 and the Internet infrastructure 3. The WebTV™ server 5 includes a proxy cache 65 which is functionally coupled to the WebTV™ client 1. The proxy cache 65 is used for temporary storage of Web documents, images, and other information which is frequently used by either the WebTV™ client 1 or the WebTV™ server 5.

A document transcoder 66 is functionally coupled between the proxy cache 65 and the Internet infrastructure 3. The document transcoder 66 includes software which is used to automatically revise the code of Web documents retrieved from the remote servers 4, for purposes which are described below.

The WebTV™ service provides a document database 61 and a user database 62, as illustrated in FIG. 4B. The user database 62 contains information that is used to control certain features relating to access privileges and capabilities of the user of the client 1. This information is used to regulate initial access to the WebTV™ service, as well as to regulate access to the individual services provided by the WebTV™ system, as will be described below. The document database 61 is a persistent database which stores certain diagnostic and historical information about each document and image retrieved by the server 5, as is now described.

A. Document Database

The basic purpose of the document database 61 is that, after a document has once been retrieved by the server 5, the stored information can be used by the server 5 to speed up processing and downloading of that document in response to all future requests for that document. In addition, the transcoding functions and various other functions of the WebTV™ service are facilitated by making use of the information stored in the document database 61, as will be described below.

Referring now to FIG. 5, the server 5 initially receives a document request from a client 1 (step 501). The document request will generally result from the user of the client 1

6

activating a hypertext anchor (link) on a Web page. The act of activating a hypertext anchor may consist of clicking on underlined text in a displayed Web page using a mouse, for example. The document request will typically (but not always) include the URL (Uniform Resource Locator) or other address of the selected anchor. Upon receiving the document request, the server 5 optionally accesses the document database 61 to retrieve stored information relating to the requested document (step 502). It should be noted that the document database 61 is not necessarily accessed in every case. The information retrieved from the document database 61 is used by the server 5 for determining, among other things, how long a requested document has been cached and/or whether the document is still valid. The criteria for determining validity of the stored document are discussed below. The server 5 retrieves the document from the cache 65 if the stored document is valid; otherwise, the server 5 retrieves the document from the appropriate remote server 4 (step 503). The server 5 automatically transcodes the document as necessary based on the information stored in the document database 61 (step 503). The transcoding functions are discussed further below.

The document database 61 includes certain historical and diagnostic information for every Web page that is accessed at any time by a WebTV™ client 1. As is well known, a Web page may correspond to a document written in a language such as HTML (Hypertext Mark-Up Language), VRML (Virtual Reality Modelling Language), or another suitable language. Alternatively, a Web page may represent an image, or a document which references one or more images. According to the present invention, once a document or image is retrieved by the WebTV™ server 5 from a remote server 4 for the first time, detailed information on this document or image is stored permanently in the document database 61. More specifically, for every Web page that is retrieved from a remote server 4, any or all of the following data are stored in the document database 61:

- 1) information identifying bugs (errors) or quirks in the Web page, or undesirable effects caused when the Web page is displayed by a client 1;
- 2) relevant bug-finding algorithms;
- 3) the date and time the Web page was last retrieved;
- 4) the date and time the Web page was most recently altered by the author;
- 5) a checksum for determining whether the Web page has been altered;
- 6) the size of the Web page (in terms of memory);
- 7) the type of Web page (e.g., HTML document, image, etc.);
- 8) a list of hypertext anchors (links) in the Web page and corresponding URLs;
- 9) a list of the most popular anchors based on the number of "hits" (requests from a client 1);
- 10) a list of related Web pages which can be prefetched;
- 11) whether the Web page has been redirected to another remote server 4;
- 12) a redirect address (if appropriate);
- 13) whether the redirect (if any) is temporary or permanent, and if permanent, the duration of the redirect;
- 14) if the Web page is an image, the size of the image in terms of both physical dimensions and memory space;
- 15) the sizes of in-line images (images displayed in text) referenced by the document defining the Web page;

7

- 16) the size of the largest image referenced by the document;
- 17) information identifying any image maps in the Web page;
- 18) whether to resize any images corresponding to the Web page;
- 19) an indication of any forms or tables in the Web page;
- 20) any unknown protocols;
- 21) any links to "dead" Web pages (i.e., pages which are no longer active);
- 22) the latency and throughput of the remote server 4 on which the Web page is located;
- 23) the character set of the document;
- 24) the vendor of the remote server 4 on which the Web page is located;
- 25) the geographic location of the remote server 4 on which the Web page is located;
- 26) the number of other Web pages which reference the subject Web page;
- 27) the compression algorithm used by the image or document;
- 28) the compression algorithm chosen by the transcoder;
- 29) a value indicating the popularity of the Web page based on the number of hits by clients; and
- 30) a value indicating the popularity of other Web pages which reference the subject Web page.

B. Transcoding

As mentioned above, the WebTV™ services provide a transcoder 66, which is used to rewrite certain portions of the code in an HTML document for various purposes. These purposes include: (1) correcting bugs in documents; (2) correcting undesirable effects which occur when a document is displayed by the client 1; (3) improving the efficiency of transmission of documents from the server 5 to the client 1; (4) matching hardware decompression technology within the client 1; (5) resizing images to fit on the television set 12; (6) converting documents into other formats to provide compatibility; (7) reducing latency experienced by a client 1 when displaying a Web page with in-line images (images displayed in text); and, (8) altering documents to fit into smaller memory spaces.

There are three transcoding modes used by the transcoder 66: (1) streaming, (2) buffered, and (3) deferred. Streaming transcoding refers to the transcoding of documents on a line-by-line basis as they are retrieved from a remote server 4 and downloaded to the client 1 (i.e., transcoding "on the fly"). Some documents, however, must first be buffered in the WebTV™ server 5 before transcoding and downloading them to the client 1. A document may need to be buffered before transmitting it to the client 1 if the type of changes to be made can only be made after the entire document has been retrieved from the remote server 4. Because the process of retrieving and downloading a document to the client 1 increases latency and decreases throughput, it is not desirable to buffer all documents. Therefore, the transcoder 66 accesses and uses information in the document database 61 relating to the requested document to first determine whether a requested document must be buffered for purposes of transcoding, before the document is retrieved from the remote server 4.

In the deferred mode, transcoding is deferred until after a requested document has been downloaded to a client 1. The deferred mode therefore reduces latency experienced by the

8

client 1 in receiving the document. Transcoding may be performed immediately after downloading or any time thereafter. For example, it may be convenient to perform transcoding during periods of low usage of WebTV™ services, such as at night. This mode is useful for certain types of transcoding which are not mandatory.

1. Transcoding for Bugs and Quirks

One characteristic of some prior art Web browsers is that they may experience failures ("crashes") because of bugs or unexpected features ("quirks") that are present in a Web document. Alternatively, quirks in a document may cause an undesirable result, even though the client does not crash. Therefore, the transcoding feature of the present invention provides a means for correcting certain bugs and quirks in a Web document. To be corrected by the transcoder 66, bugs and quirks must be identifiable by software running on the server 5. Consequently, the transcoder 66 will generally only correct conditions which have been previously discovered, such as those discovered during testing or reported by users. Once a bug or quirk is discovered, however, algorithms are added to the transcoder 66 to both detect the bug or quirk in the future in any Web document and to automatically correct it.

There are countless possibilities of bugs or quirks which might be encountered in a Web document. Therefore, no attempt will be made herein to provide an exhaustive list. Nonetheless, some examples may be useful at this point. Consider, for example, an HTML document that is downloaded from a remote server 4 and which contains a table having a width specified in the document as "0." This condition might cause a failure if the client were to attempt to display the document as written. This situation therefore, can be detected and corrected by the transcoder 66. Another example is a quirk in the document which causes quotations to be terminated with too many quotation marks. Once the quirk is first detected and an algorithm is written to recognize it, the transcoder 66 can automatically correct the quirk in any document.

If a given Web document has previously been retrieved by the server 5, there will be information regarding that document available in the document database 61 as described above. The information regarding this document will include whether or not the document included any bugs or quirks that required transcoding when the document was previously retrieved. The transcoder 66 utilizes this information to determine whether (1) the document is free of bugs and quirks, (2) the document has bugs or quirks which can be remedied by transcoding on the fly, or (3) the document has bugs or quirks which cannot be corrected on the fly (i.e., buffering is required).

FIG. 6 illustrates a routine for transcoding a Web document for purposes of eliminating bugs and quirks. Initially, the server 5 receives a document request from the client 1 (step 601). Next, the document database 61 is accessed to determine whether or not the requested document has been previously retrieved (step 602). If the document has not been previously retrieved, then the server 5 retrieves the document from the remote server 4 (step 609). Next, the retrieved document is analyzed for the presence of bugs or unusual conditions (step 610). Various diagnostic information is then stored in the document database 61 as a result of the analysis to note any bugs or quirks that were found (step 611). If any bugs or quirks were found which can be corrected by the transcoder 66, the document is then transcoded and saved to the proxy cache 65 (step 612). The transcoded document is

9

then downloaded to the client 1 (step 613). It should be noted that transcoding can be deferred until after the document has been downloaded, as described above; hence, the sequence of FIG. 6 is illustrative only.

If (in step 602) the requested document had been previously retrieved, then it is determined whether the requested document is still valid (step 603) and whether the document is present in the proxy cache 65 (step 604). If the document is no longer valid, then the document is retrieved from the remote server 4, analyzed for bugs and quirks, transcoded as required, and then downloaded to the client 1 as described above (steps 609-613). Methods for determining validity of a document are discussed below. If the document is still valid (step 603) and the document is present in the cache 65, the document is downloaded to the client 1 in its current form (as it is stored in the cache), since it has already been transcoded (step 608).

The document, however, may be valid but not present in the cache. This may be the case, for example, if the document has not been requested recently and the cache 65 has become too full to retain the requested document. In that case, the document is retrieved again from the remote server 4 (step 605) and then transcoded on the basis of the previously-acquired diagnostic information stored within the database 61 for that document. The document is then saved to the cache 65 (step 606). Note that because the document is still valid, it is assumed that the diagnostic information stored in the document database 61 for that document is still valid and that the transcoding can be performed on the basis of that information. Accordingly, once the document is transcoded, the transcoded document is downloaded to the client 1 (step 607). Again, note that transcoding can be deferred until after the document has been downloaded in some cases.

The validity of the requested document can be determined based on various different criteria. For example, some HTML documents specify a date on which the document was created, a length of time for which the document will be valid, or both. The validity determination can be based upon such information. For example, a document which specifies only the date of creation can be automatically deemed invalid after a predetermined period of time has passed.

Alternatively, validity can be based upon the popularity of the requested document. "Popularity" can be quantified based upon the number of hits for that document, which is tracked in the document database 61. For example, it might be prudent to simply assign a relatively short period of validity to a document which is very popular and a longer period of validity to a document which is less popular.

Another alternative basis for the validity of a document is the observed rate of change of the document. Again, data in the persistent document database 61 can be used. That is, because the document database 61 stores the date and time on which the document was last observed to change, the server 5 can approximate how often the document actually changes. A document or image which is observed to change frequently (e.g., a weather map or a news page) can be assigned a relatively short period of validity. It will be recognized that numerous other ways of determining validity are possible.

2. Transcoding to Reduce Latency

Another purpose for transcoding is to allow documents requested by a client 1 to be displayed by the client 1 more rapidly. Many HTML documents contain references to "in-line" images, or images that will be displayed in text in a

10

Web page. The normal process used in the prior art to display a Web page having in-line images is that the HTML document referencing the image is first downloaded to the client, followed by the client's requesting the referenced image. The referenced image is then retrieved from the remote server on which it is located and downloaded to the client. One problem associated with the prior art, however, is that the speed with which a complete Web page can be displayed to the user is often limited by the time it takes to retrieve in-line images. One reason for this is that it simply takes time to retrieve the image itself after the referencing document has been retrieved. Another reason is that, in the prior art, if the referencing document does not specify the size of the image, the Web page generally cannot be displayed until the image itself has been retrieved. The present invention overcomes these limitations.

According to the present invention, information stored in the document database 61 regarding the in-line images is used to transcode the referencing document in order to reduce latency in displaying the Web page. Once any document which references an in-line image is initially retrieved by the server 5, the fact that the document references an in-line image is stored in the document database 61. In addition, the size of the image is determined, either from the document (if specified) or from the image itself, and then stored in the document database 61. Consequently, for documents which do not specify the size of their in-line images, the size information stored in the database 61 is then used the next time the document is requested in order to reduce latency in downloading and displaying the Web page.

Refer now to FIG. 7, which illustrates a routine for reducing latency when downloading a document referencing an image to a client 1. Assume that a client 1 sends a request to the server 5 for an HTML document containing a reference to an in-line image. Assume further that the size of the image is not specified in the document itself. Initially, the server 5 determines whether that document has been previously retrieved (step 701). If not, the standard initial retrieval and transcoding procedure is followed (step 706), as described in connection with FIG. 6. If, however, the document has been previously retrieved, then the transcoder 66 accesses the size information stored in the document database 61 for the in-line image (step 702). Based on this size information, the HTML document is transcoded such that, when the Web page is initially displayed by the client 1, the area in which the image belongs is replaced by a blank region enveloping the shape of the image (step 703). Thus, any in-line image referenced by a document is displayed initially as a blank region. Consequently, the client 1 can immediately display the Web page corresponding to the HTML document even before the referenced image has been retrieved or downloaded (i.e., even before the size of the image is known to the client 1).

As the transcoded HTML document is downloaded to the client, the image is retrieved from the appropriate remote server 4 (step 704). Once the image is retrieved from the remote server 4 and downloaded to the client 1, the client 1 replaces the blank area in the Web page with the actual image (step 705).

3. Transcoding to Display Web Pages on a Television

As noted above, the client 1 utilizes an ordinary television set 12 as a display device. However, images in Web pages are generally formatted for display on a computer monitor, not a television set. Consequently, the transcoding function

11

of the present invention is used to resize images for display on the television set 12. This includes resealing images as necessary to avoid truncation when displayed on the television set 12.

It should be noted that prior art Web browsers which operate on computer monitors typically use resizable windows. Hence, the size of the visible region varies from client to client. However, because the web browser used by the WebTV™ client 1 is specifically designed for display on a television set, the present invention allows documents and images to be formatted when they are cached.

4. Transcoding for Transmission Efficiency

Documents retrieved by the server 5 are also transcoded to improve transmission efficiency. In particular, documents can be transcoded in order to reduce high frequency components in order to reduce interface flicker when they are displayed on a television set. Various methods for coding software or hardware to reduce perceptual interface flicker are described in co-pending U.S. patent application Ser. No. 08/656,923, filed on Jun. 3, 1996.

Documents can also be transcoded in order to lower the resolution of the displayed Web page. Reducing the resolution is desirable, because images formatted for computer systems will generally have a higher resolution than the NTSC (National Television Standards Committee) video format used by conventional television sets. Since the NTSC video does not have the bandwidth to reproduce the resolution of computer-formatted images, the bandwidth consumed in transmitting images to the client 1 at such a high resolution would be wasted.

5. Other Uses for Transcoding

Transcoding is also used by the present invention to recode a document using new formats into older, compatible formats. Images are often displayed in the JPEG (Joint Picture Experts Group) format or the GIF image format. JPEG often consumes less bandwidth than GIF, however. Consequently, images which are retrieved in GIF format are sometimes transcoded into JPEG format. Methods for generally converting images between GIF and JPEG formats are well known.

Other uses for transcoding include transcoding audio files. For example, audio may be transcoded into different formats in order to achieve a desired balance between memory usage, sound quality, and data transfer rate. In addition, audio may be transcoded from a file format (e.g., an ".AU" file) to a streaming format (e.g., MPEG 1 audio). Yet another use of audio transcoding is the transcoding of MIDI (Musical Instrument Digital Interface) data to streaming variants of MIDI.

Additionally, documents or images requiring a large amount of memory (e.g., long lists) can be transcoded in order to consume less memory space in the client 1. This may involve, for example, separating a large document or image into multiple sections. For example, the server 5 can insert tags at appropriate locations in the original document so that the document appears to the client 1 as multiple Web pages. Hence, while viewing a given page representing a portion of the original document, the user can view the next page (i.e., the next portion of the original document) by activating a button on the screen as if it were an ordinary hypertext anchor.

C. Proxying

As noted above, the server 5 functions as a proxy on behalf of the client 1 for purposes of accessing the Web. The

12

document database 61 is used in various ways to facilitate this proxy role, as will now be described.

1. Updating Cached Documents

It is desirable to store frequently-requested HTML documents and images in the proxy cache 65 to further reduce latency in providing Web pages to the client 1. However, because some documents and images change over time, documents in the cache 65 will not be valid indefinitely, as mentioned above. A weather map or a news-related Web page, for example, are likely to be updated quite frequently. Consequently, it is desirable for the server 5 to have the ability to estimate the frequency with which documents change, in order to determine how long a document can safely remain within the proxy cache 65 without being updated.

The persistent database 65 is used to store the date and time of the last several fetches of each document and image retrieved from a remote server 4, along with an indication of any changes that were detected, if any. A document or image which has been stored in the cache 65 is then retrieved on a periodic basis to determine if it has been changed. Change status information indicating whether the document has changed since the previous fetch is then stored in the document database 61. If no changes are detected, then the time interval between fetches of this document is increased. If the document has changed, the time interval is maintained or decreased. As a result, items in the cache 65 which change frequently will be automatically updated at frequent intervals, whereas documents which do not change often will be replaced in the cache less frequently.

FIG. 8 illustrates a routine for updating documents stored in the proxy cache 65 using data stored in the document database 61. Assume a document X has been stored in the proxy cache 65. Document X remains in the cache 65 until a predetermined update period T_1 expires (step 801). Upon the expiration of the update period T_1 , the document X is again retrieved from the appropriate remote server 4 (step 802). The newly-retrieved document X is then compared to the cached version of document X (step 803). If the document has changed, then the cached version of document X is replaced with the newly-retrieved version of document X (step 806). If not, then the update period T_1 is increased according to a predetermined time increment ΔT_1 (step 804). In any case, the date and time and the change status of document X is saved to the document database 61 (step 805).

2. Document and Image Prefetching

The document database 61 is also used by the server 5 to store prefetching information relating to documents and images. In particular, the database stores, for each document that has been retrieved, a list of images referenced by the document, if any, and their locations. Consequently, the next time a document is requested by a client 1, the images can be immediately retrieved by the server 5 (from the cache 65, if available, or from the remote server 4), even before the client 1 requests them. This procedure improves the speed with which requested Web pages are downloaded to the client.

The document database 61 is also used to facilitate a process referred to as "server-advised client prefetching." Server-advised client prefetching allows the server 5 to inform the client 1 of documents or images which are popular to allow the client 1 to perform the prefetching. In particular, for any given document, a list is maintained in the

13

server 5 of the most popular hypertext anchors in that document (i.e., those which have previously received a large number of hits). When that document is requested by the client 1, the server 5 provides the client 1 with an indication of these popular links.

3. Redirects

Web pages are sometimes forwarded from the remote server on which they are initially placed to a different location. Under the HTTP (Hypertext Transport Protocol), such forwarding is sometimes referred to as a "redirect." When an HTML document is initially stored on one remote server and then later transferred to another remote server, the first remote server will provide, in response to a request for that document, an indication that the document has been transferred to a new remote server. This indication generally includes a forwarding address ("redirect address"), which is generally a URL.

In the prior art, when a computer requesting a Web page receives a redirect, it must then submit a new request to the redirect address. Having to submit a second request and wait for a second response consumes time and increases overall latency. Consequently, the present invention uses the document database 61 to store any redirect address for each document or image. Any time a redirected document is requested, the server 5 automatically accesses the redirect address to retrieve the document. The document or image is provided to the client 1 based on only a single request from the client 1. The change in location of the redirected document or image remains completely transparent to the client 1.

FIG. 9 illustrates a routine performed by the server 5 in accessing documents which may have been forwarded to a new remote server. Initially, the server 5 receives a request for a document, which generally includes an address (step 901). The server 5 then accesses the document database 61 to determine whether there is a redirect address for the requested document (step 902). If there is no redirect address, then the server 5 accesses a remote server 4 based on the address provided in the document request from the client 1 (step 903). Assuming that the remote server 4 does not respond to the server 5 with a redirect (step 904), the document is retrieved and downloaded to the client 1 by the server 5 (step 907). If, however, a redirect address was stored in the document database 61 (step 902), then the server 5 accesses the requested document according to the redirect address (step 906). Or, if the remote server 4 responded with a redirect (step 904), then the server 5 saves the redirect address to the document database 61 (step 905) and accesses the requested document according to the redirect address (step 906).

4. Other Proxy Functions

The document database 61 also stores information relating to the performance of each remote server 4 from which a document is retrieved. This information includes the latency and throughput of the remote server 4. Such information can be valuable in instances where a remote server 4 has a history of responding slowly. For example, when the document is requested, this knowledge can be used by the server 5 to provide a predefined signal to the client 1. The client 1 can, in response to the signal, indicate to the user that a delay is likely and give the user the option of canceling the request.

5. Backoff Mode

Although the server 5 generally operates in the proxy mode, it can also enter a "backoff mode" in which the server

14

5 does not act as a proxy, or the server 5 performs only certain aspects of the normal proxying functions. For example, if the proxy cache 65 is overloaded, then the server 5 can enter a backoff mode in which documents are not cached but are transcoded as required. Alternatively, during times when the server 5 is severely overloaded with network traffic, the server 5 may instruct the client 1 to bypass the server 5 and contact remote servers 4 directly for a specified time or until further notice. Or, the server 5 can enter a flexible backoff mode in which the client 1 will be instructed to contact a remote server 4 directly only for certain Web sites for a limited period of time.

D. Access to WebTV™ Services

The WebTV™ server 5 provides various services to the client 1, such as proxying and electronic mail ("e-mail"). In the prior art, certain difficulties are associated with allowing a client computer access to different services of an Internet service, as will now be explained with reference to FIG. 10.

FIG. 10 illustrates a client-server system according to one prior art embodiment. The server 76 provides various services A, B, and C. The server 76 includes a database 71 for storing information on the user's access privileges to services A, B, and C. The client 75 of the embodiment of FIG. 10 accesses any of services A, B, and C by contacting that service directly. The contacted service then accesses the database 71, which stores the access privileges of the client 75, to determine whether the client 75 should be allowed to access that service. Hence, each service provided by the server 76 requires direct access to the database 71. This architecture results in a large number of accesses being made to the database 71, which is undesirable. In addition, the fact that each service independently has access to the database 71 raises security concerns. Specifically, it can be difficult to isolate sensitive user information. The present invention overcomes such difficulties using a technique which is now described.

1. Tickets Containing Privileges And Capabilities

As shown in FIG. 11, the server 5 provides a number of services D, E, and F 77, 79, and 80, respectively, and a log-in service 78. The log-in service 78 is used specifically to control initial log-on procedures by a client 1. The log-in service 78 has exclusive access to the user database 62 (discussed above with respect to FIG. 4B). The log-in service 78 and the user database 62 are located within a first security zone 84. Service D is located within a second security zone 86, while services E and F are contained within a third security zone 88. Note that the specific arrangement of security zones 84, 86, and 88 with respect to services D, E, and F is illustrative only.

The user database 62 of the present invention stores various information pertaining to each authorized user of a client 1. This information includes account information, a list of the WebTV™ services that are available to the particular user, and certain user preferences. For example, a particular user may not wish his client 1 to be used to access Web pages having adult-oriented subject matter. Consequently, the user would request that his account be filtered to prevent access to such material. This request would then be stored as part of the user data in the user database 62.

With regard to user preferences, the hypertext links selected by a given user can be tracked, and those having the largest number can be stored in the user database 62. The list can then be provided to the client 1 for use in generating a

15

menu screen of the user's favorite Web sites, to allow the user to directly access those Web sites. The list can also be used by the server 5 to analyze the user's interests and to formulate and provide to the user a list of new Web sites which the user is likely to be interested in. The list might be composed by associated key words in Web pages selected by the user with other Web pages.

Referring again to FIG. 11, in response to a log-on request by a client 1, the log-in service 78 consults the user database 62 to determine if access to the server 5 by this particular client 1 is authorized. Assuming access is authorized, the log-in service 78 retrieves certain user information pertaining to this particular client 1 from the user database 62. The log-in service then generates a "ticket" 82, which is an information packet including the retrieved information. The ticket 82 is then provided to the client 1 which requested access.

The ticket 82 includes all information necessary to describe the access privileges of a particular user with respect to all services provided by the server 5. For example, the ticket may include the user name registered to the client 1, the e-mail address assigned to client 1, and any filtering requested by the user with respect to viewing Web sites. Each time the user requests access to one of the services D, E, or F, the client 1 submits a copy of the ticket 82 to that service. The requested service can then determine from the copy of the ticket 82 whether access to that service by that client 1 is authorized and, if so, any important information relating to such access.

None of the services provided by the server 5, other than the log-in service 78, has access to the user database 62. Hence, any security-sensitive information can be isolated within the user database 62 and the log-in service 78. Such isolation allows the individual services provided by the server 5 to be placed within separate "firewalls" (security regions), illustrated as security zones 84, 86, and 88. In addition, this technique greatly reduces the number of accesses required to the user database 62 compared to the prior art embodiment illustrated in FIG. 10.

2. Redundancy of Services and Load Balancing

The present invention also includes certain redundancies in the various services provided by the server 5. In particular, a given service (e.g., e-mail) can be provided by more than one physical or logical device. Each such device is considered a "provider" of that service. If a given provider is overloaded, or if the client 1 is unable to contact that provider, the client 1 can contact any of the other providers of that service. When the server 5 receives a log-in request from a client 1, in addition to generating the above-described ticket 82, the log-in service 78 dynamically generates a list of available WebTV™ services and provides this list to the client 1.

The server 5 can update the list of services used by any client 1 to reflect services becoming unavailable or services coming on-line. Also, the list of services provided to each client 1 can be updated by the server 5 based upon changes in the loading of the server 5, in order to optimize traffic on the server 5. In addition, a client's list of services can be updated by services other than the log-in service 78, such that one service can effectively introduce another service to the client 1. For example, the e-mail service may provide a client 1 with the name, port number and IP of its address book service. Thus, one service can effectively, and securely within the same chain of trust, introduce another service to the client 1.

16

This list of services includes the name of each service, a port number for the provider of each service, and an IP (Internet Protocol) for each service. Different providers of the same service are designated by the same name, but different port numbers and/or IPs. Note that in a standard URL, the protocol is normally specified at the beginning of the URL, such as "HTTP://www. . . ." under the HTTP protocol. However, according to the present invention, the normal protocol designation (i.e., "HTTP") in the URL is replaced with the name of the service, since the port number and IP for each service are known to the client 1. Hence, the client 1 can access any of the redundant providers of a given service using the same URL. This procedure effectively adds a level of indirection to all accesses made to any WebTV™ service and automatically adds redundancy to the proxy service. It should also be noted that separate service names can also refer to the same service.

Assume, for example, that the e-mail service provided by the WebTV™ system is designated by the service name "WTV-mailto." A client 1 can access any provider of this e-mail service using the same URL. The client 1 merely chooses the appropriate port number and IP number to distinguish between providers. If the client 1 is unable to connect to one e-mail provider, it can simply contact the next one in the list.

Thus, at log-in time, a client 1 is provided with both a ticket containing privileges and capabilities as well as a list of service providers, as illustrated in FIG. 12. Initially, the log-in service 78 determines whether the user of client 1 is a valid user (step 1201). If not, log-in is denied (step 1205). If the user is a valid user, then the log-in service 78 gathers user information from the user database 62 and generates a ticket 82 (step 1202). The log-in service 78 also generates the above-described list of services (step 1203). The ticket 82 and the list of services are then downloaded to the client 1 (step 1204).

3. Asynchronous Notification to Clients by Server

Another limitation associated with prior art Internet servers is the inability to provide asynchronous notification information to the client in the absence of a request from the client to do so. It would be desirable, for example, for a server to notify a client on its own initiative when a particular Web page has changed or that a particular service is inaccessible. The server 5 of the present invention provides such capability, and the client 1 is configured to receive and decode such notifications. For example, the client 1 can receive updates of its listing of service providers from the server 5 at various points in time, as already described. Similarly, if a particular service provider becomes unavailable, that fact will be automatically communicated to the client 1. As another example, if e-mail addressed to the user has been received by the server 5, then the server 5 will send a message to the client 1 indicating this fact. The client 1 will then notify the user that e-mail is waiting by a message displayed on the television set 12 or by an LED (light emitting diode) built into the housing of WebTV™ box 10.

Thus, a method and apparatus have been described for providing proxying and transcoding of documents in a network. The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

17

What is claimed and desired to be secured by United States Letters Patent is:

1. In a proxying server coupled to a client and to a remote server, the proxying server operating as a proxy on behalf of the client for accessing the remote server, a method of retrieving and transcoding a document requested by the client, the method comprising the steps of:

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document in order to perform at least one function selected from the following functions:

matching decompression requirements at the client; converting the document into a format compatible for the client;

reducing latency experienced by the client; and altering the document to fit into smaller memory space; and

transmitting the document to the client.

2. A method according to claim 1, wherein the request includes a URL (Uniform Resource Locator).

3. A method according to claim 1, wherein the step of transcoding is performed while the step of retrieving the document is performed.

4. A method according to claim 1, wherein the step of transcoding comprises the step of reading at least a portion of the document from a proxy cache located in the proxying server.

5. A method according to claim 1, wherein the step of transmitting the document to the client is performed prior to performing the step of transcoding, at the proxying server, the data in the document.

6. A method according to claim 1, wherein the step of transmitting the document to the client comprises the step of transmitting the transcoded document to the client, wherein the step of transmitting the document to the client is performed after the step of transcoding, at the proxying server, the data in the document.

7. A method according to claim 1, wherein the document includes a link to another document, the link including a retrieval address, and wherein the step of transcoding at the proxying server the data in the document comprises the step of updating the link

8. A method according to claim 7, wherein the other document is an image, and wherein the step of updating the link comprises the step of adding information to the document indicating the size of the image.

9. A method according to claim 8, wherein the other document is inaccessible to the proxying server, and wherein the step of updating the link comprises the step of removing the link.

10. A method according to claim 7, wherein the other document has been relocated from the retrieval address to a redirect address, and wherein the step of updating the link comprises the step of updating the link to correspond to the redirect address.

11. A method according to claim 1, wherein the client includes a television display, wherein the document references an image, and wherein the step of transcoding, at the proxying server, the data in the document is conducted in order to perform at least the function of converting the document into a compatible format for the client, and comprises the step of revising the data such that the image is sized for display on the television display.

18

12. A method according to claim 1, wherein the document references an image having a first image format, and wherein the step of transcoding, at the proxying server, the document is conducted in order to perform at least the function of converting the document into a compatible format for the client, and includes the step of converting the first image format to a second image format.

13. A method according to claim 1, further comprising the steps of:

identifying an image referenced by the document;

determining whether the image has been previously retrieved by the proxying server; and

if the image has been previously retrieved by the proxying server, accessing information stored in the proxying server indicating the size of the image;

wherein the step of transcoding, at the proxying server, the data in the document is conducted in order to perform at least the function of reducing latency experienced by the client, and comprises the step of using the information indicating the size of the image to revise the data of the document to allow the document to be partially displayed by the client before the image is received by the client.

14. A method according to claim 13, wherein the step of transcoding, at the proxying server, the data in the document comprises the following step:

if the image has been previously retrieved by the proxying server, transcoding at the proxying server the data in the document without the image but providing a space for the image to be later inserted; and

wherein the step of transmitting the document to the client comprises the following steps:

transmitting the document to the client without the image but providing the space for the image to be later inserted; and

after transmitting the document to the client without the image, transmitting the image to the client.

15. In a proxying server coupled to a client and to a remote server, the proxying server operating as a proxy on behalf of the client for accessing the remote server, a method of retrieving and transcoding a document requested by the client, the method comprising the steps of:

providing a persistent database at the proxying server, the persistent database including information relating to the document;

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document using the information included in the persistent database in order to perform at least one function selected from the following functions:

matching decompression requirements at the client; converting the document into a format compatible for the client;

reducing latency experienced by the client; and altering the document to fit into smaller memory space; and

transmitting the document to the client.

16. A method according to claim 15, further comprising the step of storing in the persistent database validity information corresponding to the document.

19

17. A method according to claim 16, wherein the step of retrieving the document comprises the following steps:

retrieving the document from the persistent database if the validity information corresponding to the document indicates that the document is valid; and

retrieving the document from the remote server if the validity information corresponding to the document indicates that the document is not valid.

18. A method according to claim 16, wherein the step of storing in the persistent database validity information corresponding to the document comprises the step of observing a rate of change of the document.

19. A method according to claim 18, wherein the step of observing a frequency of change of the document comprises the step of periodically retrieving the document, wherein successive retrievals of the document are separated in time by at least a time interval.

20. A method according to claim 19, wherein the step of periodically retrieving the document comprises the following steps:

retrieving the document at a first time;

storing the document retrieved at the first time in a memory;

retrieving the document at a second time, the second time being at least the time interval after the first time;

determining whether the document retrieved at the second time has changed compared to the document retrieved at the first time;

if the document retrieved at the second time is different than the document retrieved at the first time, replacing the document retrieved at the first time with the document retrieved at the second time in the memory; and

if the document retrieved at the second time is the same as the document retrieved at the first time, extending the time interval.

21. A method according to claim 15, further comprising the step of storing in the persistent database performance information relating to performance of the remote server when accessing the document.

22. A method according to claim 21, wherein the performance information comprises a latency value.

23. A method according to claim 15, further comprising the step of storing in the persistent database information for optimizing memory usage by the client.

24. A method according to claim 15, wherein the step of retrieving the document comprises the following steps:

determining whether a redirect address corresponding to the document is stored in the persistent database;

if the redirect address corresponding to the document is stored in the persistent database, accessing the remote server using the redirect address, the remote server corresponding to the redirect address; and

if the redirect address corresponding to the document is not stored in the persistent database, the method further comprises the following steps:

accessing a previous remote server at which the document previously resided;

obtaining the redirect address from the previous remote server;

storing the redirect address in the persistent database; and

20

accessing the remote server using the redirect address.

25. A computer program product for implementing, in a proxying server coupled to a client and to a remote server, a method of retrieving and transcoding a document requested by the client, the computer program product comprising a computer-readable medium carrying computer-executable instructions for causing the proxying server to perform acts included in the method, said acts comprising:

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document in order to perform at least one function selected from the following functions:

matching decompression requirements at the client;

converting the document into a format compatible for the client;

reducing latency experienced by the client; and
altering the document to fit into smaller memory space; and

transmitting the document to the client.

26. A computer program product according to claim 25, wherein the computer-executable instructions comprise a plurality of program code means for transcoding at least a portion of the document, including:

program code means for transcoding at least a portion of the document so as to match decompression requirements at the client;

program code means for transcoding at least a portion of the document so as to convert the document into a format compatible with the client;

program code means for transcoding at least a portion of the document so as to reduce latency experienced by the client; and

program code means for transcoding at least a portion of the document so as to alter the document to fit into smaller memory space.

27. A computer program product according to claim 26, wherein the act of transcoding comprises selecting at least one of said plurality of program code means for transcoding for use in the act of transcoding.

28. A computer program product according to claim 25, wherein the act of transcoding is performed while the act of retrieving the document is performed.

29. A computer program product according to claim 25, wherein the act of transcoding comprises reading at least a portion of the document from a proxy cache located in the proxying server.

30. A computer program product according to claim 25, wherein the act of transmitting the document to the client is performed prior to performing the act of transcoding.

31. A computer program product according to claim 25, wherein the act of transmitting the document to the client comprises the act of transmitting the transcoded document to the client, wherein the act of transmitting the document to the client is performed after the act of transcoding.

* * * * *



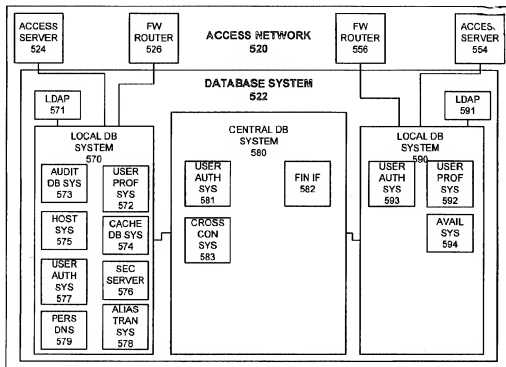
US006697806B1

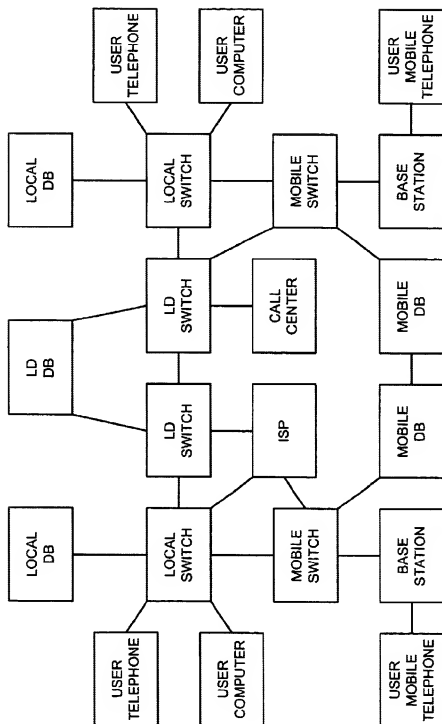
**(12) United States Patent
Cook****(10) Patent No.: US 6,697,806 B1
(45) Date of Patent: Feb. 24, 2004****(54) ACCESS NETWORK AUTHORIZATION****(75) Inventor: Fred S. Cook, Olathe, KS (US)****(73) Assignee: Sprint Communications Company,
L.P., Overland Park, KS (US)****(*) Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.**(21) Appl. No.: 09/574,978****(22) Filed: May 19, 2000****Related U.S. Application Data****(63)** Continuation of application No. 09/556,276, filed on Apr. 24, 2000.**(51) Int. Cl.⁷ G06F 17/30****(52) U.S. Cl. 707/10; 707/3; 707/9;
709/227; 709/203; 709/219****(58) Field of Search 707/1-3, 9-10,
707/4, 104.1; 709/223-227, 219, 203****(56) References Cited****U.S. PATENT DOCUMENTS**5,884,312 A * 3/1999 Dustan et al. 707/10
6,151,601 A * 11/2000 Papierniak et al. 707/1

* cited by examiner

Primary Examiner—Jean R. Homere
Assistant Examiner—Mohammad Ali**(57) ABSTRACT**

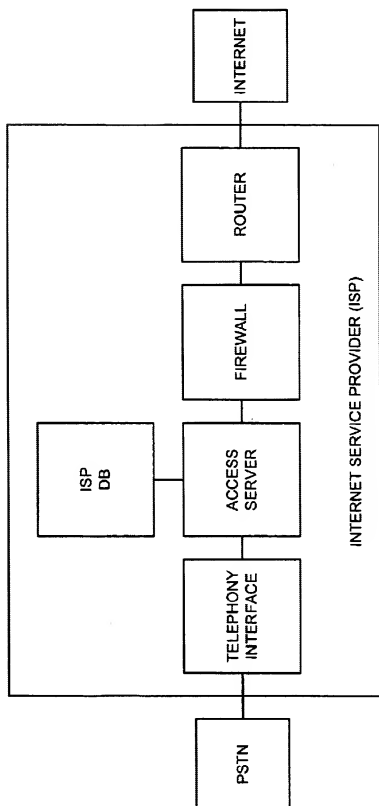
An access communication system provides access between a user system and a plurality of communication networks. The plurality of communication networks provide services to a user in the user system. An access communication system includes a local database system and an access server that is connected to the user system and the plurality of communication networks. The local database system receives a user logon. The local database system then processes the user logon to determine if the user is allowed access to the access communication system based on a local database system. The local database system then provides access to the access communication system to the user in response to the determination that the user is allowed access based on the local database system. The local database system then generates an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system. The local database system receives and processes an authorization response indicating whether the user is allowed to use the access system from the second database system. The local database system then provides access to the access communication system to the user in response to the authorization response that allows the user to use the access communication system.

24 Claims, 63 Drawing Sheets



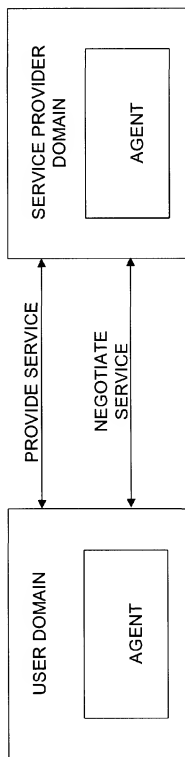
PRIOR ART

FIG. 1



PRIOR ART

FIG. 3



PRIOR ART

FIG. 4

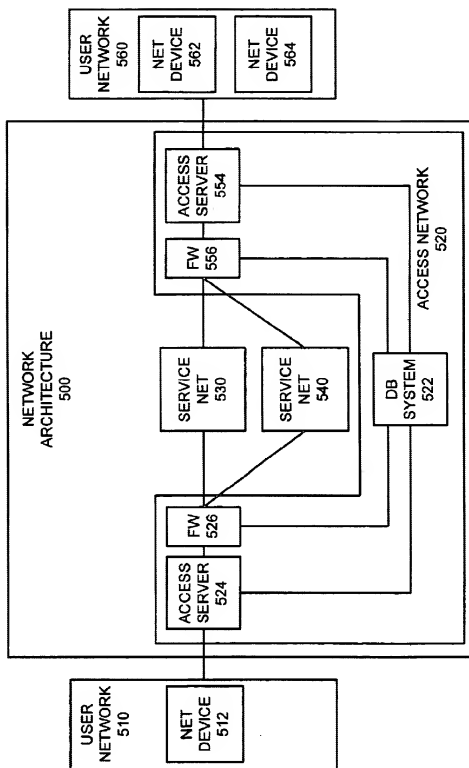


FIG. 5

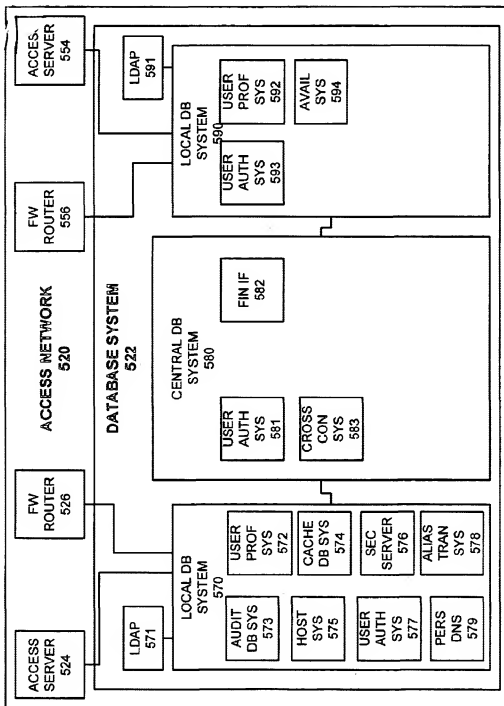


FIG. 6

USER ID	PASSWORD	NAME	ACCOUNT NUMBER	SERVICES	ADDRESS	BILLING CODE	CLASS	GROUP	SHELL	MACROS

FIG. 7

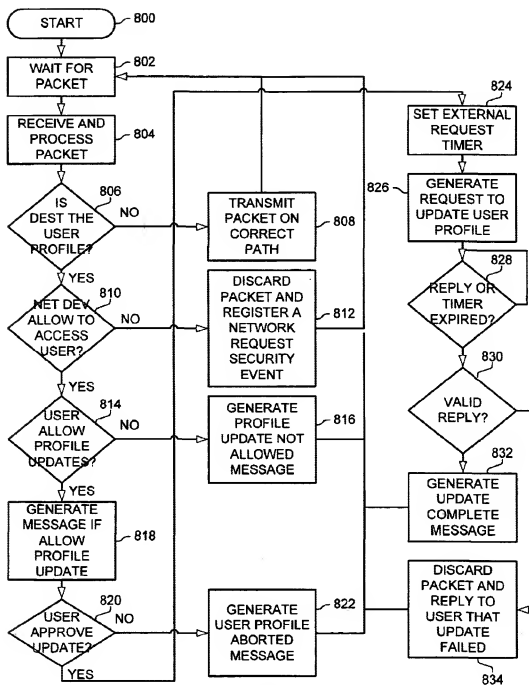


FIG. 8

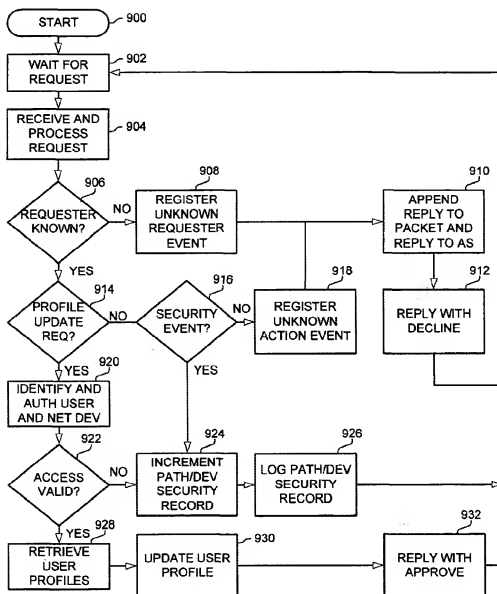


FIG. 9

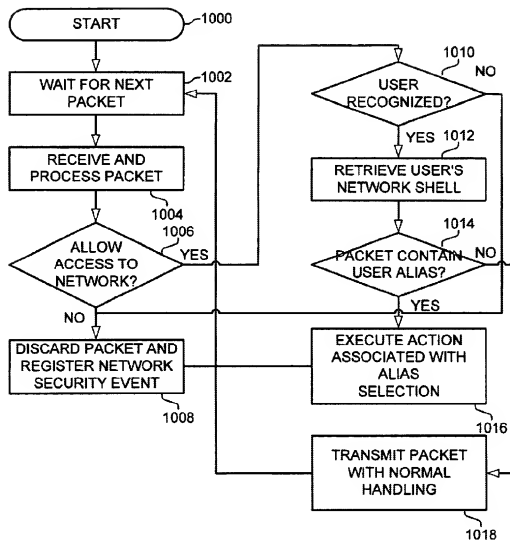


FIG. 10

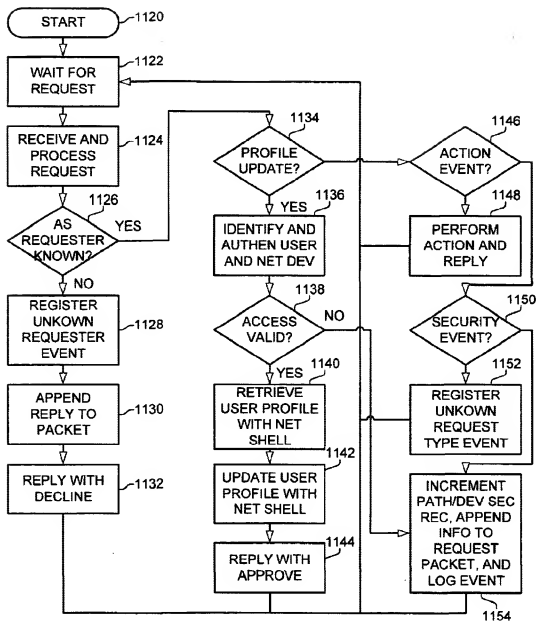


FIG. 11

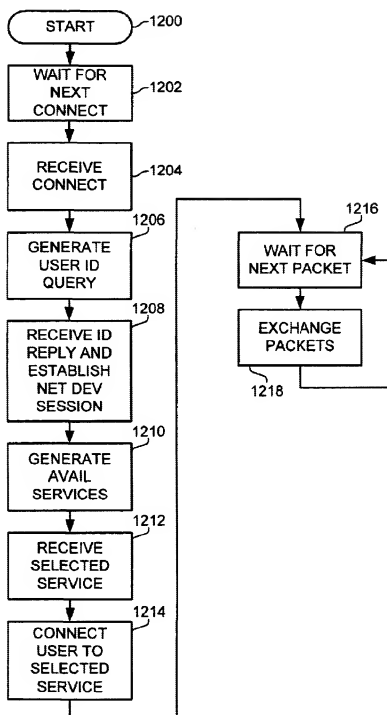


FIG. 12

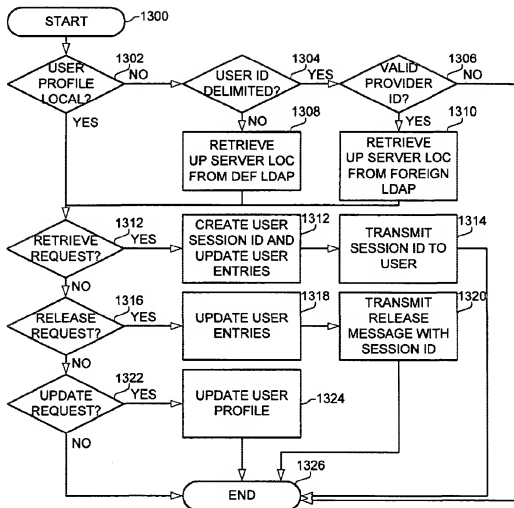


FIG. 13

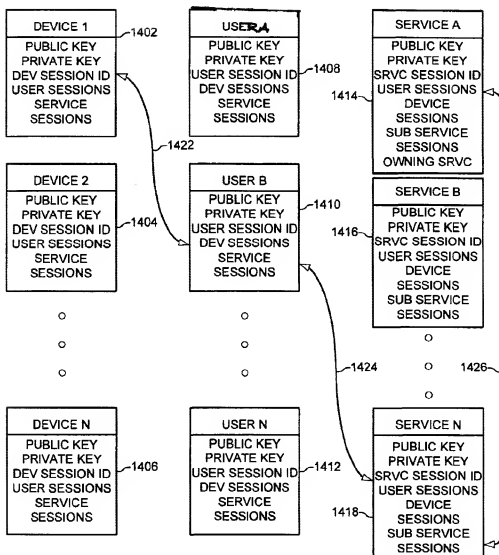
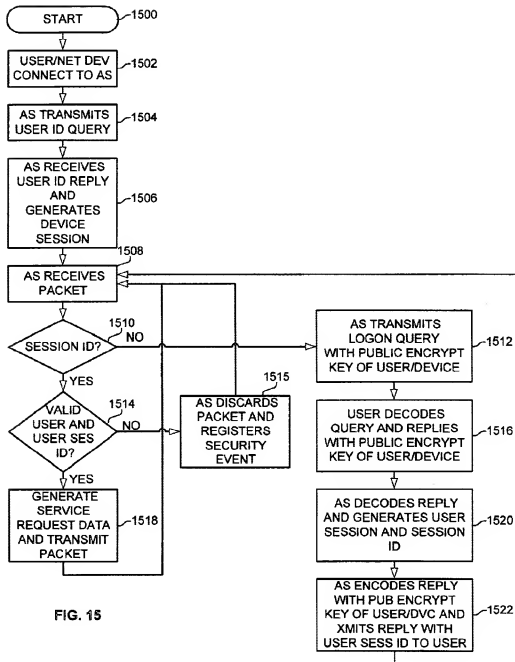


FIG. 14



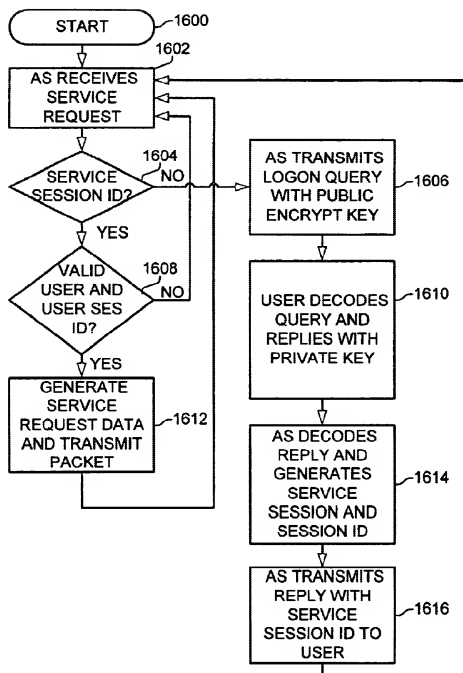


FIG. 16

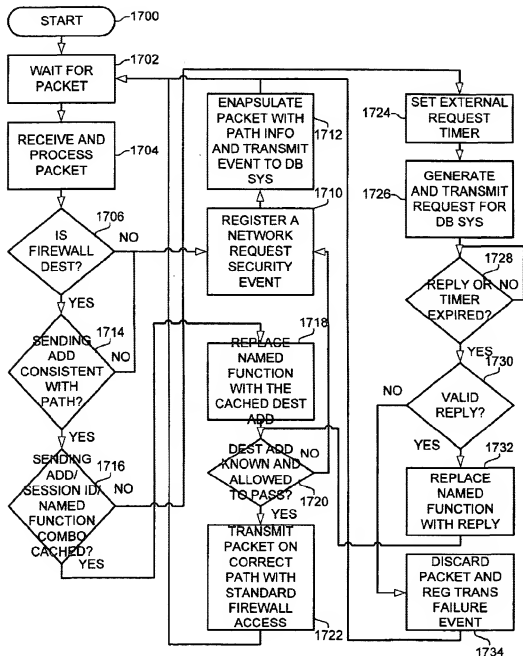


FIG. 17

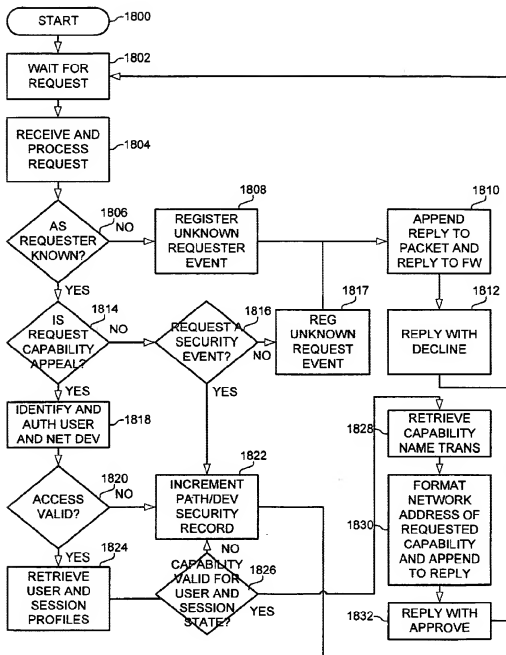


FIG. 18

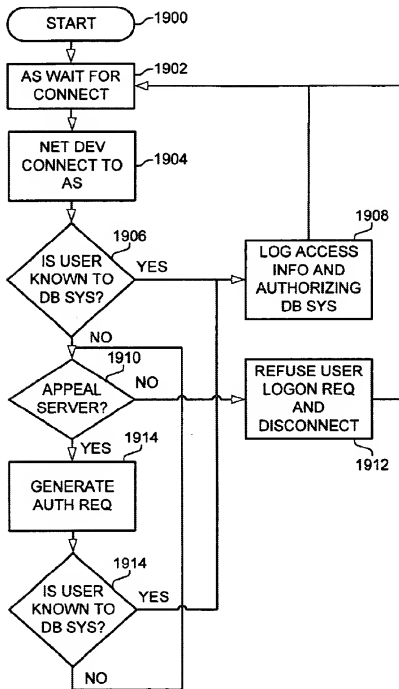


FIG. 19

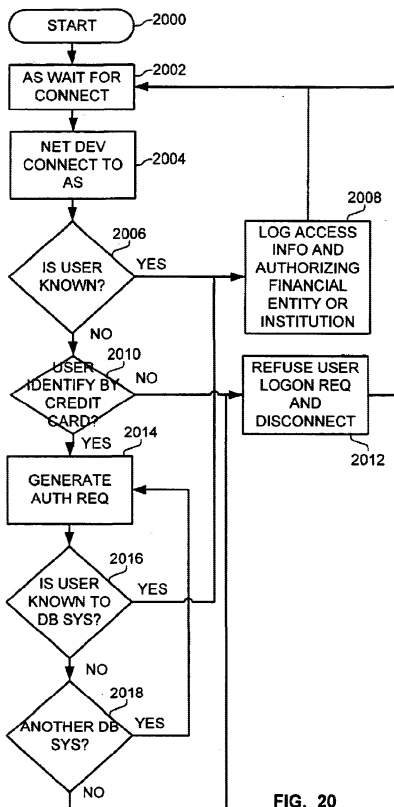


FIG. 20

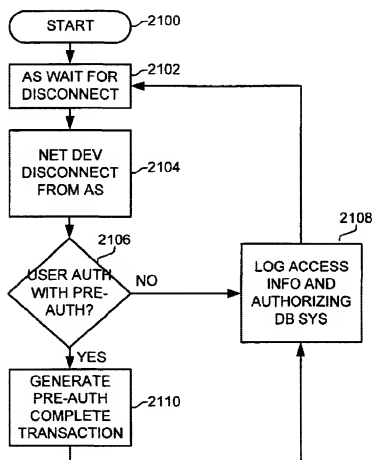


FIG. 21

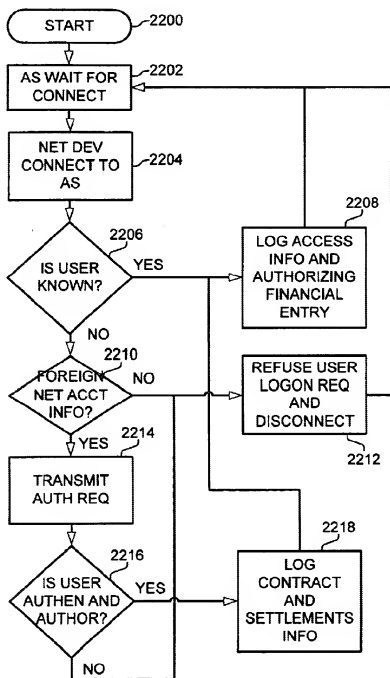
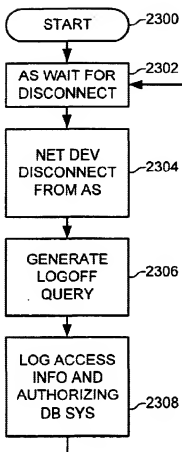


FIG. 22

**FIG. 23**

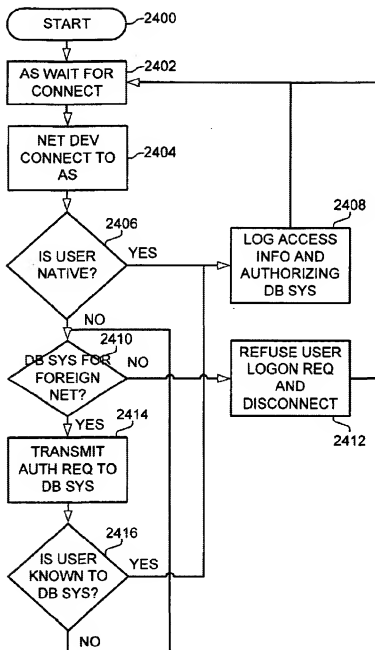


FIG. 24

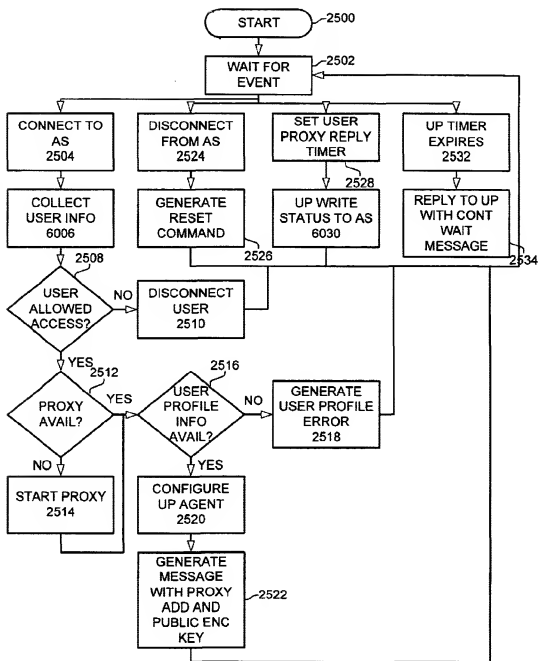


FIG. 25

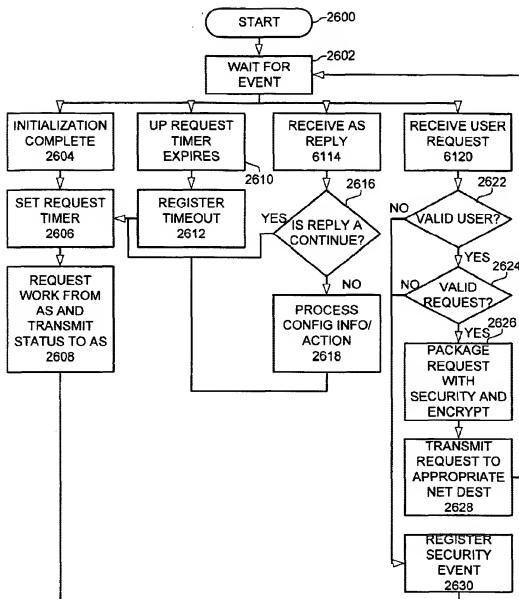


FIG. 26

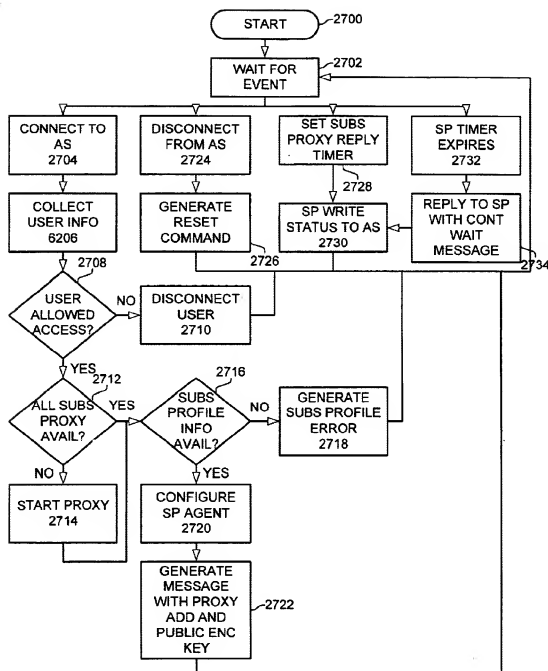


FIG. 27

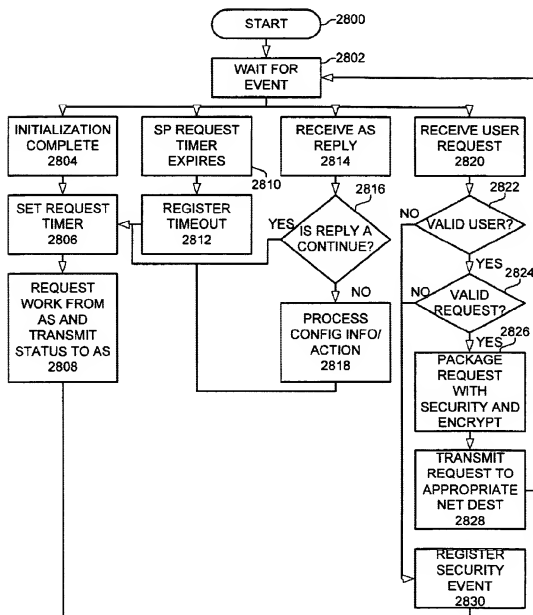


FIG. 28

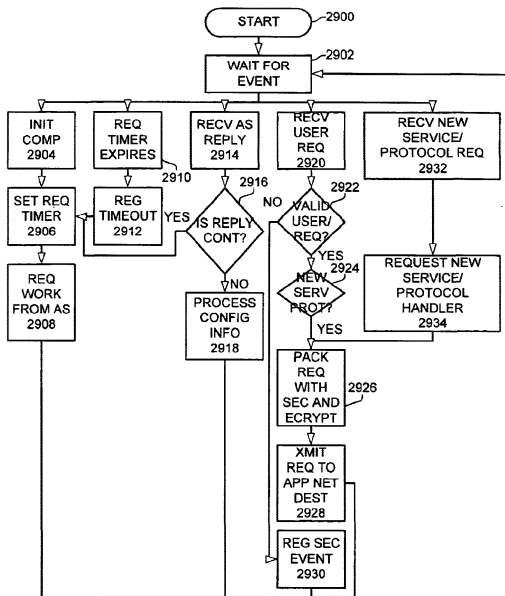


FIG. 29

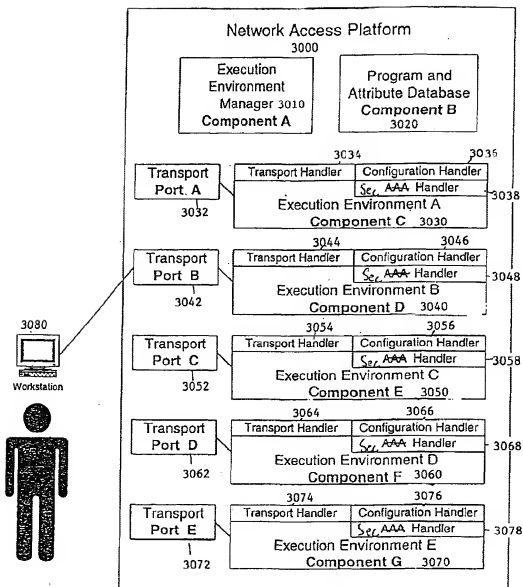


FIGURE 30

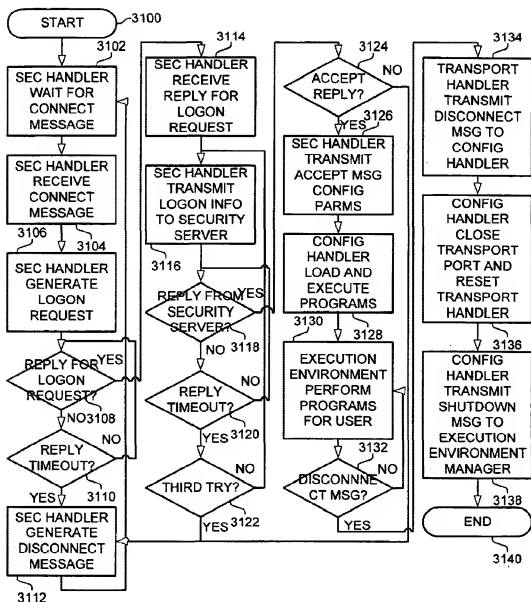


FIG. 31

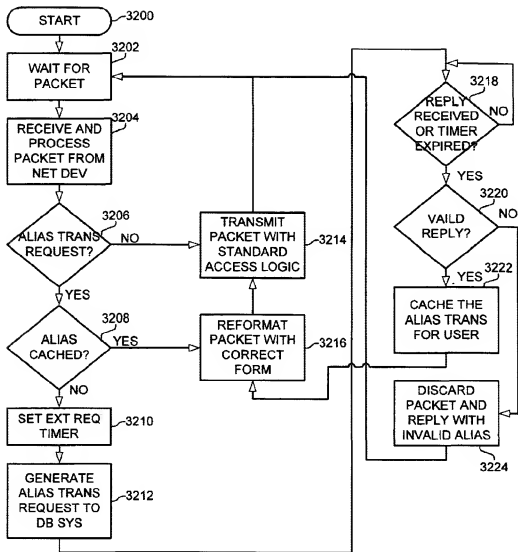


FIG. 32

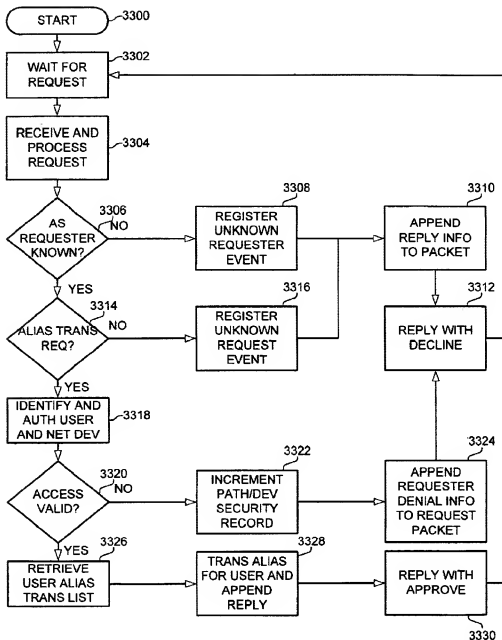


FIG. 33

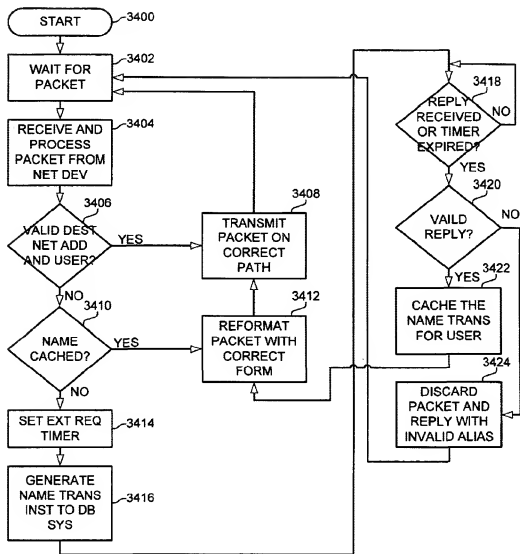


FIG. 34

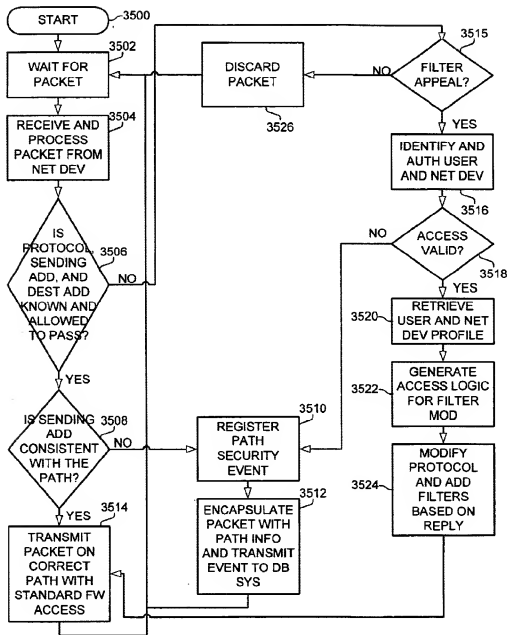


FIG. 35

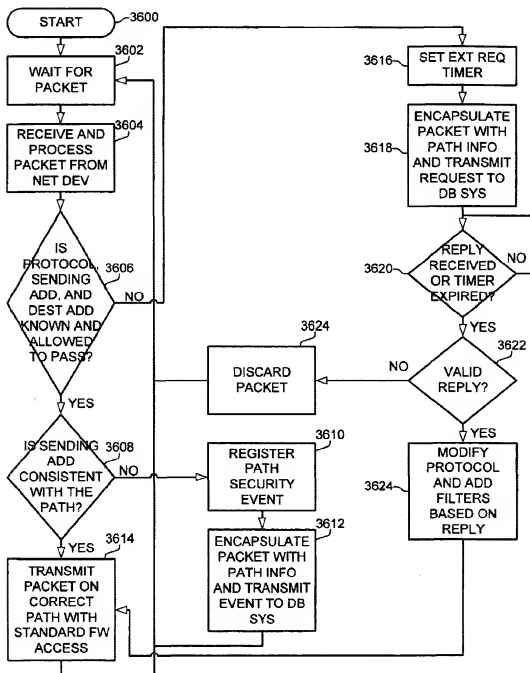


FIG. 36

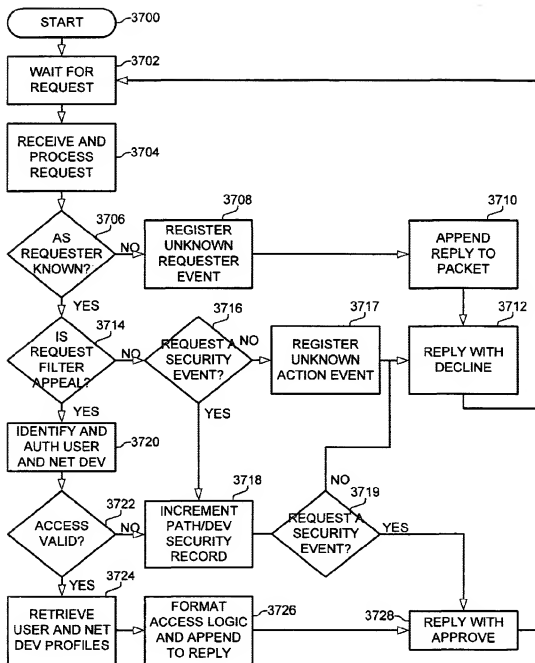


FIG. 37

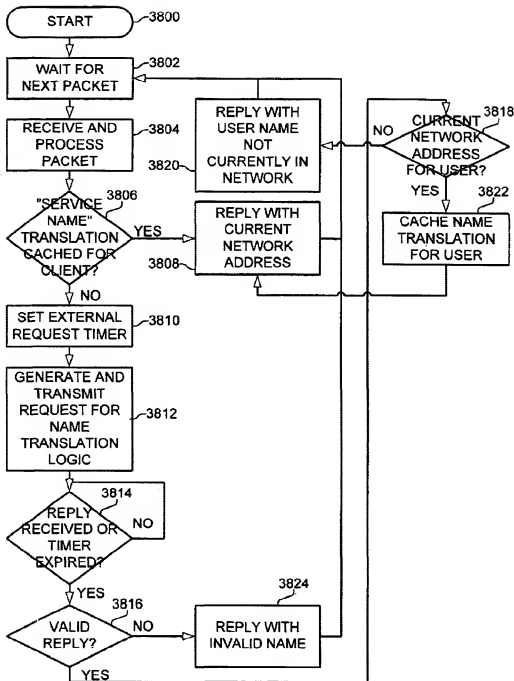


FIG. 38

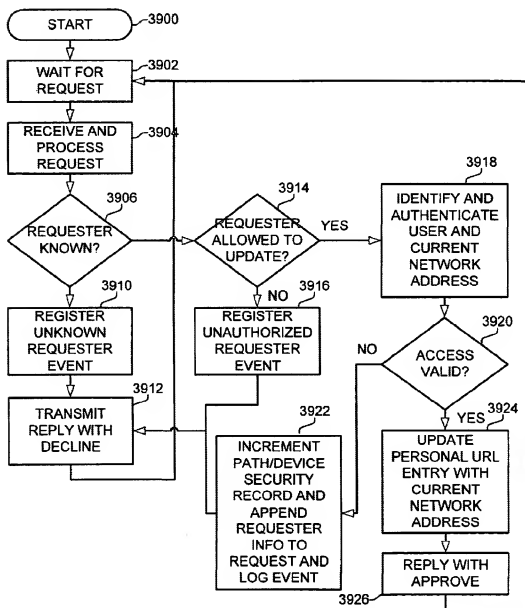


FIG. 39

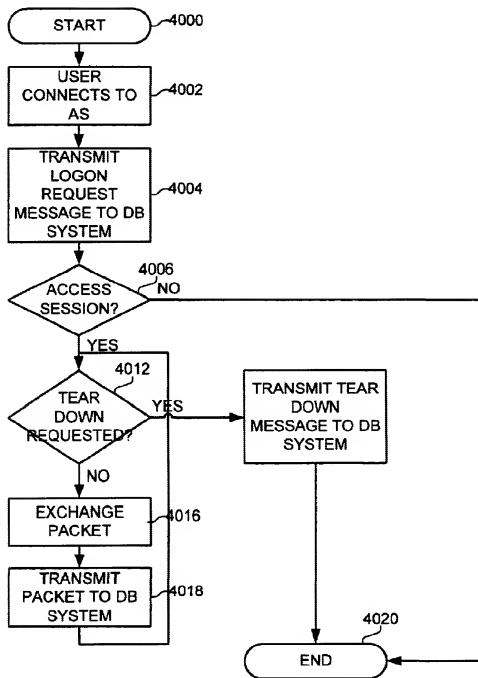


FIG. 40

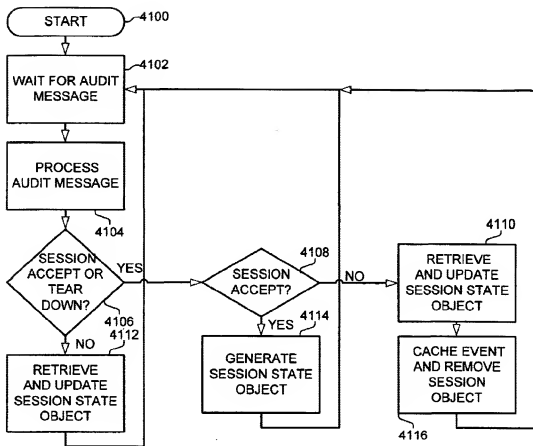


FIG. 41

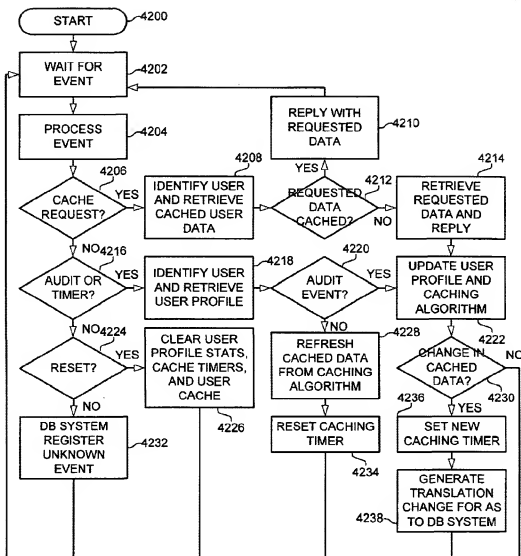


FIG. 42

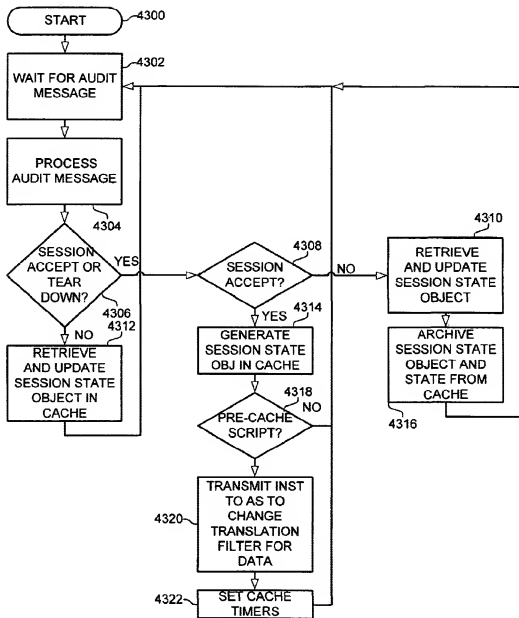


FIG. 43

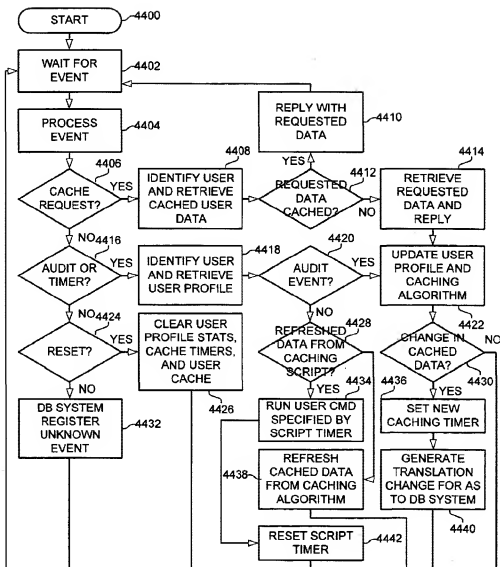


FIG. 44

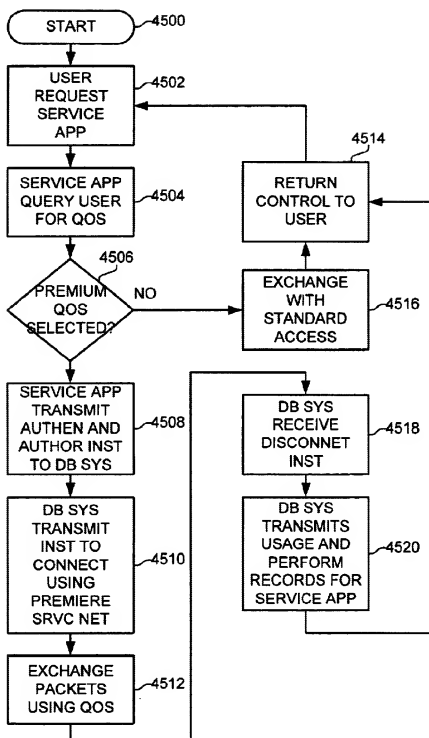


FIG. 45

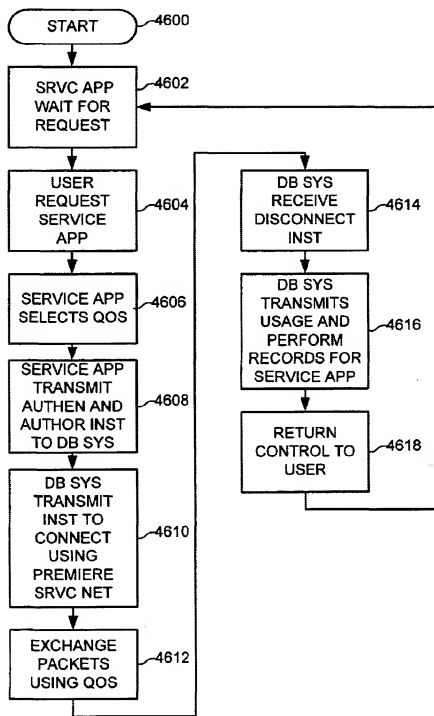


FIG. 46

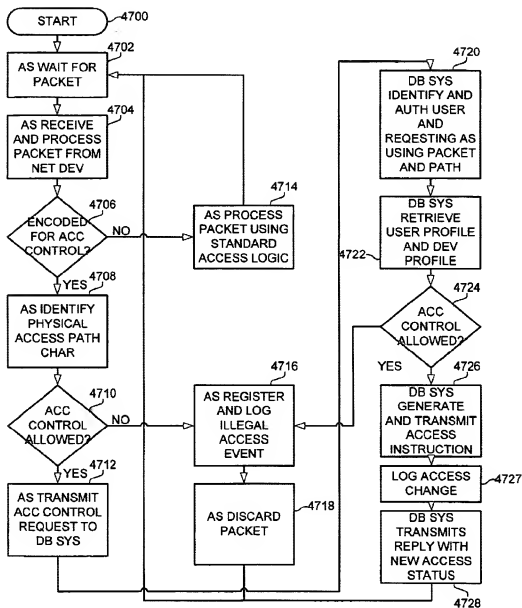


FIG. 47

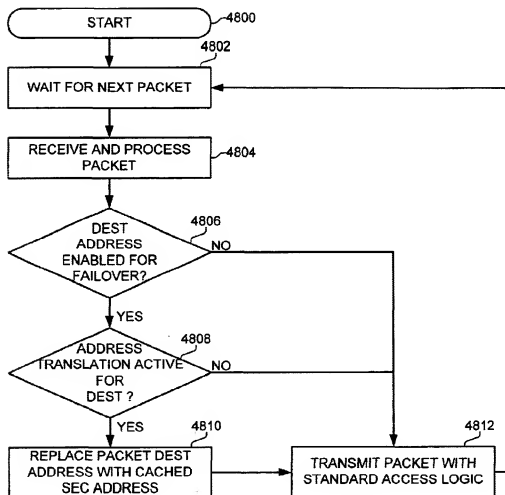


FIG. 48

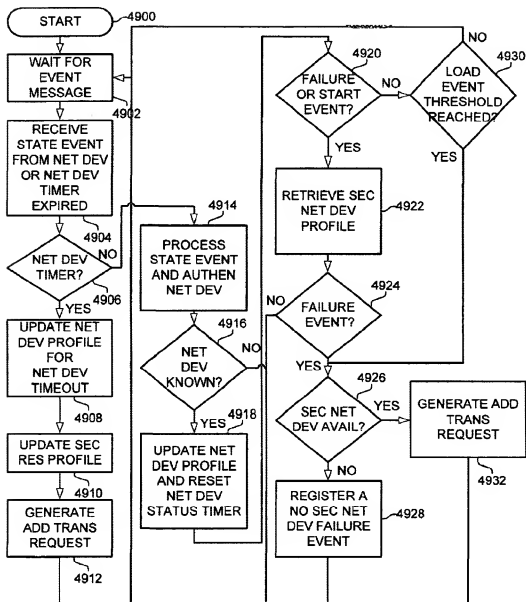


FIG. 49

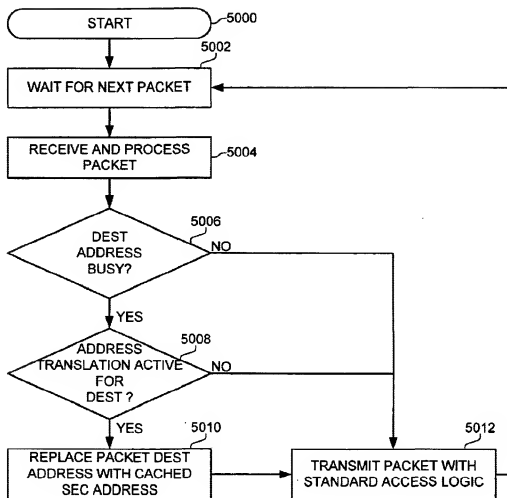


FIG. 50

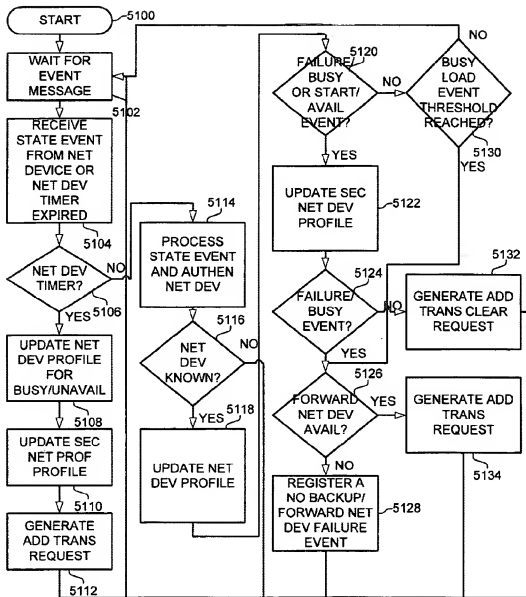


FIG. 51

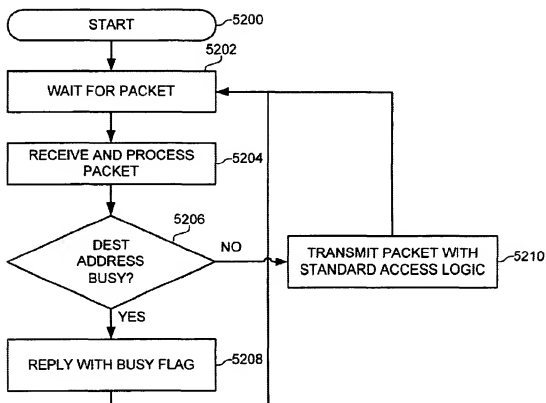


FIG. 52

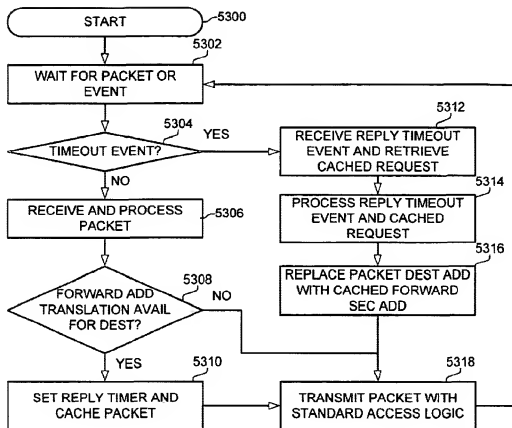


FIG. 53

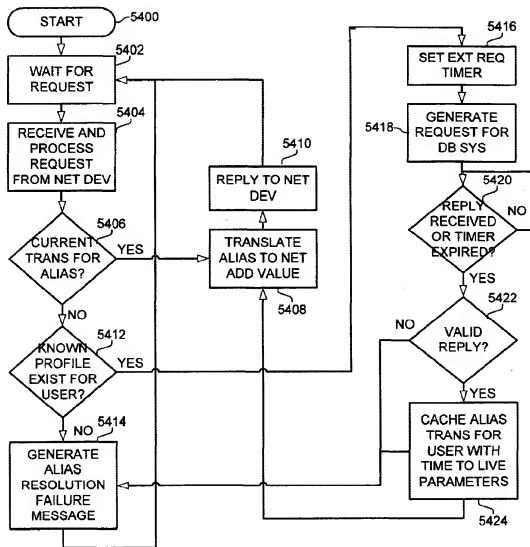


FIG. 54

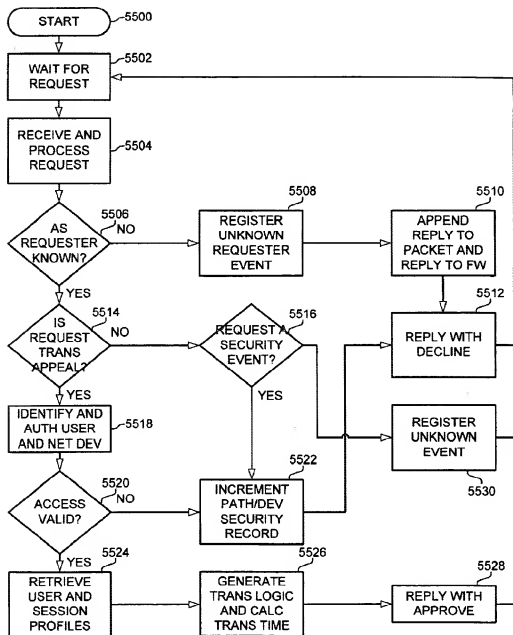


FIG. 55

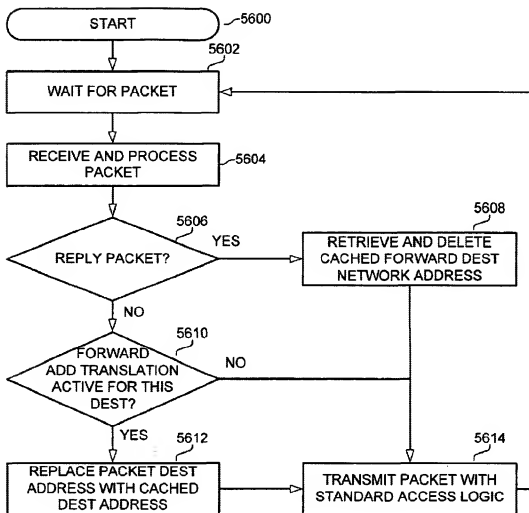


FIG. 56

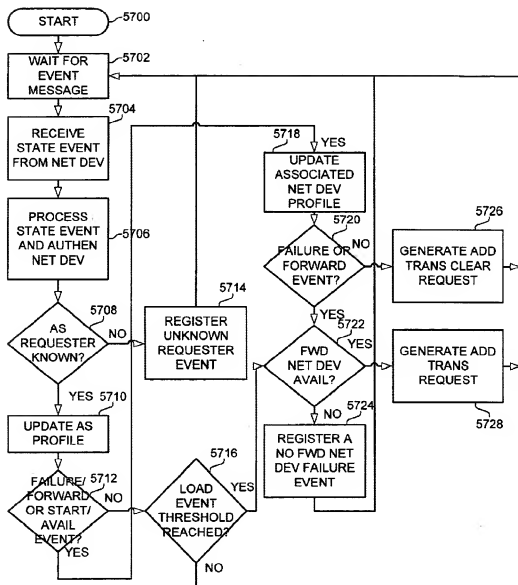


FIG. 57

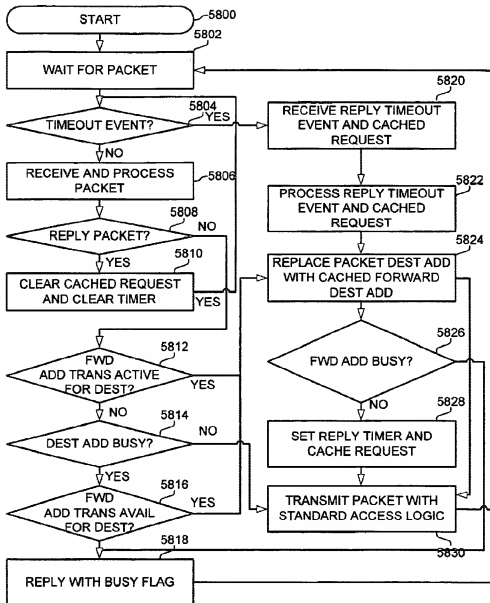


FIG. 58

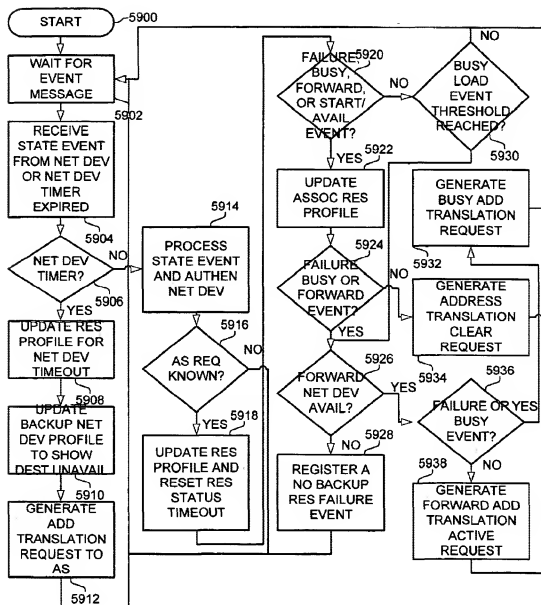
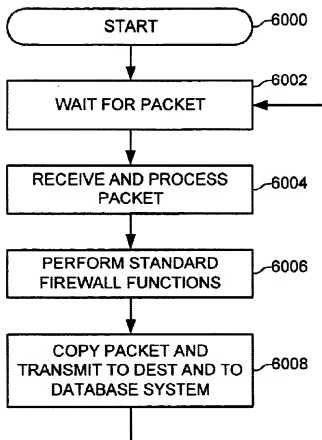


FIG. 59

**FIG. 60**

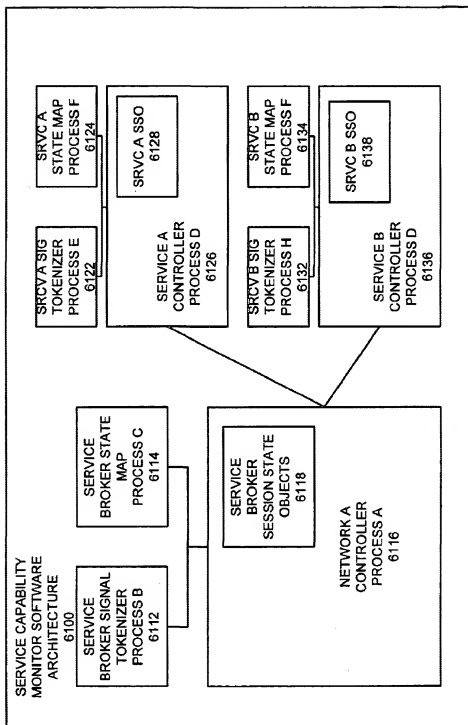
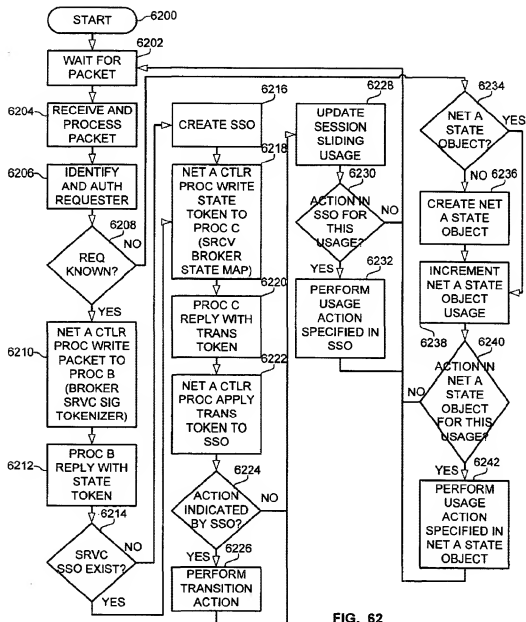


FIG. 61



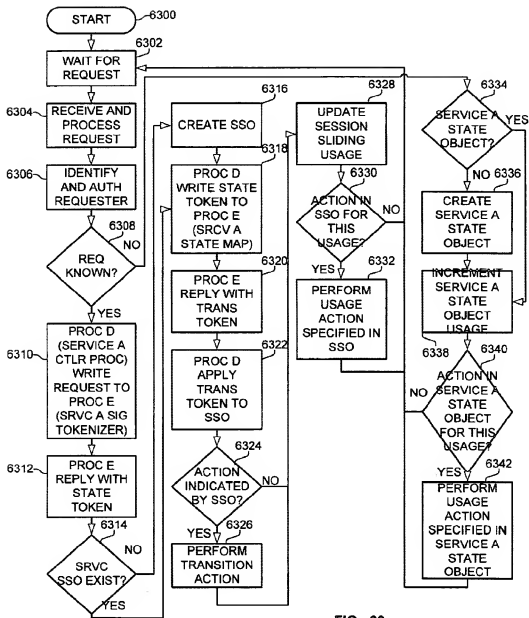


FIG. 63

ACCESS NETWORK AUTHORIZATION

RELATED APPLICATIONS

This application is a continuation of prior application Ser. No. 09/556,276, entitled "Access Communication System", filed Apr. 24, 2000, currently pending, and incorporated by reference into this application.

FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

Not applicable

MICROFICHE APPENDIX

Not applicable

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention is related to the field of communication networks, and in particular, to an access communication system that provides access to multiple service provider systems. More particularly, this invention relates to a system that authorizes access to use the access communication system.

2. Description of the Prior Art

Communication networks have seen dramatic development over the past several years so that today, there are multiple diverse communication networks providing services. The current technical challenge is to develop interfaces between the networks to provide seamless service across multiple networks. Unfortunately, today's interfaces lack the ability to offer the user with easy access to services from multiple systems. These interfaces do not customize operations for the user.

FIG. 1 illustrates the conventional public telephone network. User telephones and computers are connected to local switches. The local switches are coupled to a local database. The user places calls through the local switch. The local switch processes the called number to provide an end-point connection or access to other networks. The local switch may connect the call to another telephone in the local calling area. In a Local Number Portability (LNP) situation, the local switch exchanges information with the local database to obtain the appropriate routing number for a ported call. The local switch may also connect the call to an Internet Service Provider. If a Digital Subscriber Line (DSL) is used, DSL equipment may be used to bypass the local switch. The local switch may also connect the call to a long distance switch. To provide access to the long distance switch, the local switch first exchanges information with the local database. The local database identifies the long distance network for either the user or the dialed number.

The long distance switch processes the called number to route the call to another system or network. Prior to routing, the long distance switch validates the call by checking the caller's number. The long distance switch may also exchange information with the long distance database to provide special call-handling. One example is a calling card call, where the long distance database validates an account number for billing the call. Another example is a toll-free call, where the long distance database processes external information customized by the called party to route the call. Toll-free routing information includes items such as time and date, caller location, and call center status. Many long distance calls are simply routed through the long distance

network to another local network for call completion. Other calls are routed from the long distance network to a call center. Call centers offer a concentration of call-handling capabilities for operations, such as order entry, customer service, and promotions. Call centers include automatic call distribution equipment to route calls to the appropriate destination within the call center.

Both local and long distance networks exchange calls with mobile switches. The mobile switches are connected to base stations that communicate over the air with wireless telephones. When a mobile user places a call, the mobile switches exchange information with a mobile database to validate the mobile caller. The call is then routed to another mobile caller, the telephone network, or to an ISP. When a mobile caller moves around, their wireless phone logs-in with the physically proximate mobile network, and that mobile network updates the mobile user's location in the mobile databases. When a call is placed to that mobile user, their home mobile switch obtains a routing number from the mobile databases to route the call to the mobile network currently in communication with the mobile user.

FIG. 2 illustrates a conventional data network that transfers packets of user data to a destination based on address information carried in the packets. Users are connected to Local Area Networks (LANs) that are connected to Wide Area Networks (WANs). A common LAN is an Ethernet system. A common WAN is an intranet. WANs are interconnected by data networks, such as IP, T1, frame relay, or Asynchronous Transfer Mode (ATM). WANs are connected to the Internet through ISPs. WANs are connected to the public telephone network through telephony gateways. A common telephony gateway is a Private Branch Exchange (PBX).

FIG. 3 illustrates a conventional ISP. The public telephone network is coupled to a telephony interface that converts between telephony analog and digital protocols and the Internet Protocol (IP). Some telephony interfaces also handle DSL traffic that may already use IP. The telephony interface transfers IP traffic through an access server and firewall to a router. Some ISPs combine the firewall and the access server into one system. Also, the position of the firewall may vary, and traffic shapers may be present. The router exchanges IP traffic with the Internet.

In operation, the user calls the ISP over the telephone network and logs-in at the access server. The access server collects and forwards the user name and password to the ISP database. The ISP database validates the user name and password and returns an IP address to the access server. The IP address is for the user's terminal connection. Using the IP address, the user may communicate through the firewall to the router for transmissions to an IP address. The user now has Internet access through the router and exchanges packets with various Internet servers.

IP addresses are referred to as network addresses and include a network ID and a host ID. Network IDs are unique across the Internet and host IDs are unique within a given network. IP addresses are lengthy numerical codes, so to simplify things for the user, service addresses are available that are easier to remember. The service addresses are often the name of the business followed by ".com". Domain Name Service (DNS) is hosted by servers on the Internet and translate between service addresses and network addresses. The browser in the user computer accesses the DNS to obtain the desired network address.

FIG. 4 illustrates conventional network access. A current proposal for communication network access is provided by

3

the Telecommunication Information Network Architecture Consortium (TINA-C). TINA-C proposes the use of agents in the user domain and the service provider domain. The service provider domain could be a telephone network, data network, or ISP. The agents negotiate access service rights. Once the service is negotiated, the user receives the service from the service provider network during a service session. Unfortunately, the access session occurs between the user domain and a particular service provider domain. At present, the service provider domain provides limited access capability beyond simply handing off communications to another network based on a called number or network address. As a result, the ability to customize services for a particular user across multiple service providers is inadequate.

SUMMARY OF THE INVENTION

The inventions solve the above problems by providing access between a user system and a plurality of communication networks. The plurality of communication networks provide services to a user in the user system. An access communication system includes a database system and an access server that is connected to the user system and the plurality of communication networks.

In one aspect of the inventions for user access profile inheritance, the database system receives an update request from the access server to update a user access profile through inheritance. The database system then processes the update request to inherit user profile information from a user profile data structure. The database system updates the user access profile with the user profile information.

In another aspect of the inventions for network shells, the access server receives an alias selection from a user for a network shell that includes alias selections associated with actions. The access server then processes the alias selection to execute an action associated with the alias selection.

In another aspect of the inventions for service based directory, the access server transmits a list of services to a user system. The access server then receives a selection from the list of services. The access server processes the selection to generate an instruction to provide the service related to the selection.

In another aspect of the inventions for user access profile mobility, the database system receives user information. The database system then processes the user information to determine if a user access profile is local within a local database system. The database system generates and transmits a request to retrieve a user access profile from a second database system external to the local database system in response to the determination that the user access profile is not local.

In another aspect of the inventions for service, user, and device sessions, the access communication system establishes a connection between a network device and the access server. The access communication system then generates a device session including a device session ID based on the network device. The access communication system generates and transmits a login query for the network device. The access communication system receives and processes a login reply from the network device to generate a user session including a user session ID based on the user. The access communication system receives and processes a request for the service to generate a service session including a service session ID based on the service. The service may generate and transmit a login query for the user. The access communication system links the device session, user session, and the service session using the device session ID, the user session ID, and the service session ID.

4

In another aspect of the inventions for service capability firewall, the access server receives information including a named function request for a service provider. The access server processes the information to check if the named function request is valid for the service provider and the service. If valid, the access server determines if a private destination address exists for the named function request. The access server replaces the named function request with the private destination address in response to the determination that the private destination address exists for the named function request. The access server then transmits the information with the private destination address to the service provider.

In another aspect of the inventions for prepaid access and bank card access, the database system receives information identifying a billing code for a user. The database system then processes the billing code to determine if the user is allowed to use the access system. The database system provides access to the access system in response to the determination that the user is allowed to use the access system.

In another aspect of the inventions for global authentication and access card, the database system receives a user login. The database system then processes the user login to determine if the user is allowed access to the access communication system based on a local database system. The database system then provides access to the access communication system to the user in response to the determination that the user is allowed access based on the local database system. The database system then generates an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system. The database system receives and processes an authorization response indicating whether the user is allowed to use the access system from the second database system. The database system then provides access to the access communication system to the user in response to the authorization response that allows the user to use the access communication system.

In another aspect of the inventions for user based proxy and subscriber based proxy, the database system includes a user proxy. The user proxy receives a request for the service from the user system. The user proxy then transmits the request for the service to a service provider. The user proxy exchanges user information between the user system and the service provider.

In another aspect of the inventions for dynamic proxy, the database system includes a proxy. The proxy receives a service/protocol request for a new service or protocol. The proxy processes the service/protocol request to generate a handler request to obtain a handler for the new service or protocol. The proxy then receives and executes the handler for the new service or protocol.

In another aspect of the inventions for access execution environment, the database system receives and processes a login reply into an access execution environment for a user. The database system retrieves programs for the user into the access execution environment. The database system executes the programs for the user in the access execution environment.

In another aspect of the inventions for domain name scoping and inband domain name service lookup, the access server receives information including an alias from the user system. The access server determines if the alias exists in a cache including aliases and alias translations for the user.

5

The access server changes the information based on the cached alias translation.

In another aspect of the inventions for inline access service triggering, the access server receives information. The access server then processes the information to determine if the information is allowed to pass. The access server changes access logic based on the information in response to the determination that the information is not allowed to pass. The access server changes the filters of the access server based on the information in response to the determination that the information is not allowed to pass.

In another aspect of the inventions for access service triggering, the access server receives information. The access server processes the information to determine if the information is allowed to pass. The access server then generates a request from a database system in response to the determination that the information is not allowed to pass. The access server receives a reply including access logic from the database system. The access server changes filters of the access server based on the access logic.

In another aspect of the inventions for personal URL, the database system receives information including a user alias. The database system processes the information to determine if a user alias translation including a current network address for the user alias exists. The database system then modifies the information with the current network address using the user alias translation.

In another aspect of the inventions for predictive caching, the access server receives a request for data. The access server then determines if the data exists in a user cache wherein the user cache contains cached data based on the user's predictive patterns. The access server retrieves the data from the user cache in response to the determination that the data exists in the user cache. The access server transmits the request for data for the service provider in response to the determination that the data does not exist in the user cache.

In another aspect of the inventions for user controlled caching, the access server receives a request for data. The access server determines if the data exists in a user cache wherein the user cache contains cached data based on a user's script of commands. The access server retrieves the data from the user cache in response to the determination that the data exists in the user cache. The access server then transmits the request for data for the service provider in response to the determination that the data does not exist in the user cache.

In another aspect of the inventions for service usage audit, the access server receives an audit message into a database system. The access server processes the audit message to store the audit message in the database system.

In another aspect of the inventions for switching access by a user, switching access by a service provider, and dynamic access control, the database system receives a request. The database system processes the request to determine if the switching of the access is allowed. The database system then generates an instruction to switch access in response to the determination that the switching is allowed.

In another aspect of the inventions for network failover, network busy forwarding, time-out, busy flag, forwarding, and network endpoint availability management, the access server receives information for a destination network device. The access server determines if the destination network device is available. The access server performs an action in response to the determination that the destination network device is unavailable.

6

In another aspect of the inventions for scheduled alias translation, the access server receives information including an alias. The access server processes the information to determine whether an alias translation exists based on an alias translation schedule. The access server then modifies the information based on the alias translation in response to the determination the alias translation exists.

In another aspect of the inventions for service capability monitor, the database system receives information from an access server during a service session. The database system determines a current state of the service session based on the information. The database system determines a state transition based on the current state and a map of state transitions of the service. The database system determines whether the state transition is valid for the service session.

BRIEF DESCRIPTION OF THE DRAWINGS

The same reference number represents the same element on all drawings.

FIG. 1 illustrates a public telephone network in the prior art.

FIG. 2 illustrates a data network in the prior art.

FIG. 3 illustrates an Internet Service Provider in the prior art.

FIG. 4 illustrates conventional network access.

FIG. 5 illustrates a network architecture in an example of the invention.

FIG. 6 illustrates an access network in an example of the invention.

FIG. 7 illustrates a table for a user access profile in an example of the invention.

FIG. 8 illustrates a flowchart for an access server for inheriting a user access profile in an example of the invention.

FIG. 9 illustrates a flowchart for a database system for inheriting a user access profile in an example of the invention.

FIG. 10 illustrates a flowchart of an access server for executing a network shell in an example of the invention.

FIG. 11 illustrates a flowchart of a database system for updating a network shell in an example of the invention.

FIG. 12 illustrates a flowchart for the services based directory in an example of the invention.

FIG. 13 illustrates a flowchart of user access profile mobility in an example of the invention.

FIG. 14 illustrates a logical view of device, user, and service sessions in an example of the invention.

FIG. 15 illustrates a flowchart for a user based session in an example of the invention.

FIG. 16 illustrates a flowchart for a service based session in an example of the invention.

FIG. 17 illustrates a flowchart for a firewall/router for service capability firewall in an example of the invention.

FIG. 18 illustrates a flowchart for a database system for service capability firewall in an example of the invention.

FIG. 19 illustrates a flowchart for prepaid access in an example of the invention.

FIG. 20 illustrates a flowchart for bank card access for a connection in an example of the invention.

FIG. 21 illustrates a flowchart for network access cards for a disconnection in an example of the invention.

FIG. 22 illustrates a flowchart for network access cards for a connection in an example of the invention.

7

FIG. 23 illustrates a flow chart for network access cards for a disconnection in an example of the invention.

FIG. 24 illustrates a flowchart for global access in an example of the invention.

FIG. 25 illustrates a flowchart for an access server for user based proxies in an example of the invention.

FIG. 26 illustrates a flowchart for a user proxy for user based proxies in an example of the invention.

FIG. 27 illustrates a flowchart for an access server for subscriber based proxies in an example of the invention.

FIG. 28 illustrates a flowchart for a subscriber proxy for subscriber based proxies in an example of the invention.

FIG. 29 illustrates a flowchart for a dynamic proxy for dynamic proxies in an example of the invention.

FIG. 30 illustrates a block diagram of the access execution environment in an example of the invention.

FIG. 31 illustrates a flow chart for the access execution environment in an example of the invention.

FIG. 32 illustrates a flowchart for an access server for domain name scoping in an example of the invention.

FIG. 33 illustrates a flowchart for a database system for domain name scoping in an example of the invention.

FIG. 34 illustrates a flowchart for an access server for an inband domain name service lookup in an example of the invention.

FIG. 35 illustrates a flowchart for inline access service triggering in an example of the invention.

FIG. 36 illustrates a flowchart for an access server for access service triggering in an example of the invention.

FIG. 37 illustrates a flowchart for a database system for access service triggering in an example of the invention.

FIG. 38 illustrates a flow chart for the personal URL lookup in an example of the invention.

FIG. 39 illustrates a flow chart for the personal URL update in an example of the invention.

FIG. 40 illustrates a flowchart for an access server for auditing in an example of the invention.

FIG. 41 illustrates a flowchart for a database system for auditing in user predictive caching in an example of the invention.

FIG. 42 illustrates a flowchart for a database system for caching in user predictive caching in an example of the invention.

FIG. 43 illustrates a flowchart for a database system for auditing in user controlled caching in an example of the invention.

FIG. 44 illustrates a flowchart for a database system for caching in user controlled caching in an example of the invention.

FIG. 45 illustrates a flowchart for switching access by a user in an example of the invention.

FIG. 46 illustrates a flowchart for switching access by a service provider in an example of the invention.

FIG. 47 illustrates a flowchart for dynamic switching access in an example of the invention.

FIG. 48 illustrates a flowchart for an access server for network address failover in an example of the invention.

FIG. 49 illustrates a flowchart for a database system for network address failover in an example of the invention.

FIG. 50 illustrates a flowchart for an access server for network busy forwarding in an example of the invention.

FIG. 51 illustrates a flowchart for a database system for network busy forwarding in an example of the invention.

8

FIG. 52 illustrates a flowchart for an access server for a busy flag when the destination network device is busy in an example of the invention.

FIG. 53 illustrates a flowchart for an access server for forwarding, if the destination network device timeouts in an example of the invention.

FIG. 54 illustrates a flowchart for an access server for schedule alias resolution in an example of the invention.

FIG. 55 illustrates a flowchart for a database system for scheduled alias resolution in an example of the invention.

FIG. 56 illustrates a flowchart for an access server for destination controlled forwarding in an example of the invention.

FIG. 57 illustrates a flowchart for a database system for destination controlled forwarding in an example of the invention.

FIG. 58 illustrates a flowchart for an access server for network endpoint availability management in an example of the invention.

FIG. 59 illustrates a flowchart for a database system for network endpoint availability management in an example of the invention.

FIG. 60 illustrates a flowchart for a firewall/router for service capability monitor in an example of the invention.

FIG. 61 illustrates a service capability monitor software architecture for a service capability monitor in an example of the invention.

FIG. 62 illustrates a flowchart for the network logic for service capability monitor in an example of the invention.

FIG. 63 illustrates a flowchart for the service logic for service capability monitor in an example of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Communication Network Architecture—FIGS. 5 and 6

The following description and associated figures discuss specific examples intended to teach the present invention to those skilled in the art. Those skilled in the art will appreciate numerous variations from these examples that do not depart from the scope of the invention. Those skilled in the art will also appreciate that various features described below could be combined in various ways to form multiple variations of the invention.

FIG. 5 illustrates a network architecture 500 in an example of the invention. The network architecture 500 comprises a service network 530, a service network 540, and an access network 520. The access network 520 comprises a database system 522, an access server 524, a firewall/router 526, a firewall/router 556, and an access server 554. A user network 510 includes a network device 512. The user network 510 is connected to the access server 524. The access server 524 is connected to the firewall/router 526 and the database system 522. The firewall/router 526 is connected to the service network 530, the service network 540, and the database system 522. The firewall/router 556 is connected to the service network 530, the service network 540, the database system 522, and the access server 554. The access server 554 is connected to the database system 522 and the user network 560. The user network 560 comprises a network device 562 and a network device 564.

The access network 520 provides an interface between the user network 510 and 560 and the service networks 530 and

540. The interface function provides user access profiles, security, switching, and caching. The user network 510 and 560 could be a residential or business communication system that includes network devices. A network device 512, 562, and 564 could be any device configured to exchange data or information with the access network 520. Some examples of network devices are wireless and wireline telephones, computers, modems, servers, and/or data terminals, along with associated interconnections and network interfaces. For illustrative purposes in the examples below, a user interacts with the network device 512 to access services provided by the network architecture 500 and the user network 560. The user network 510 could also be a communication destination for other users. In these cases, the access network 520 provides destination performance monitoring and control. The example of a user interacting with the network device 562 and 564 to access services provided by the network architecture 500 and the user network 510 is not discussed below for the sake of clarity.

The access server 524 and the database system 522 could be adapted from components used in current ISPs, and the access network 520 could be integrated into these ISP components. The database system 522 houses user access profiles for the users in the user networks 510 and 560 with external access rights. The user access profiles provide a data clearinghouse for user-related information for security, service options, current state, and customized macros. The service networks 530 and 540 could be voice or data systems, such as the public telephone network, Internet, public data networks, and private data networks.

When a user requests access to services, the access network 520 processes the user access profile for the user. The access network 520 performs security measures to validate the user. The access network 520 then binds the user to a terminal and to a service. This user-terminal-service binding correlates the user's capabilities in the user access profile with the capabilities of the user's current terminal and the current capabilities of the service provider. The access network 520 activates a network shell for the user and the user's terminal. The network shell is an interface that is customized for the user and the terminal. The user invokes services using the network shell.

The user access profile may contain several macros that can be as simple as address translations or as complex as lengthy computer programs. The user's network shell provides access to the user's macros. The user access profile also stores caching instructions. The caching instructions control the collection and storage of information within the access network 520 for immediate access by the user.

FIG. 6 depicts an access network in an example of the invention. The access network 520 comprises the database system 522, the access server 524, the firewall/router 526, the firewall/router 556, and the access server 554. The database system 522 comprises a local database system 570, a Lightweight Directory Access Protocol (LDAP) interface system 571, a central database system 580, a local database system 590, and an LDAP interface system 591. The local database system comprises a user profile system 572, an audit database system 573, a cache database system 574, a host system 575, a security server 576, a user authorization system 577, an alias translation system 578, and a personal DNS system 579. The central database system 580 comprises a user authorization system 581, a financial interface 582, and a cross connect system 583. The local database system 590 comprises a user profile system 592, a user authorization system 593, and an availability system 594.

The access server 524 and the firewall/router 526 are connected to the local database system 570. The local

database system 570 is connected to the central database system 580 and the LDAP interface system 571. The central database system 580 is connected to the local database system 590. The local database system 590 is connected to the firewall/router 556, the access server 554, and the LDAP interface system 591.

An access provider is an entity that provides access to users who use communications services. A typical access provider comprises an access server, a firewall/router, and a local database system. The systems within the local database system 570 could be included within the local database system 590. Also, the systems within the local database system 590 could be included within the local database system 570. As discussed above, the user is accessing the network architecture through the user network 510. The duplication of systems in the local database system 570 and the local database system 590 is excluded for the purposes of clarity. A service provider is an entity that provides communication services to users who are accessing the service through an access provider.

Network User Access Profile—FIGS. 5, 6, 7, 8, 9, 10, 11, and 12

User Access Profile Inheritance

The user access profile is stored in the access network 520 and controls user access to services. The user access profile is any information or data associated with controlling user access to a service such as role identification, authorization, billing information and access preferences.

FIG. 7 depicts table for a user access profile in an example of the invention. A user access profile includes access information for the user, billing information, and preferences for access. Access information is any information or data related to providing the user access to the network architecture 500. Some examples of access information are user ID, password, name, account number, user alias, current network address, switching allowed flag, and other security information. Access information also include a list of services that the user has subscribed to or is allowed access to. In one embodiment, access information includes a cache of information that the user has accessed previously. Some examples of billing information are address and billing code including bank card numbers or prepaid account codes. Preferences for access allow the user to save choices or preferences to customize their access to the network architecture 400. Some examples of preferences for access are file formats and Quality of Service values. The user access profile may also include usage information such as time of day access, day of week, usages per day, usages per week, and usages per month.

Users typically set up their user access profiles when signing up for the access to the network architecture 500. In one embodiment, users create the user access profile through an inheritance process. Through the inheritance process, the user selects user profile information from other user access profiles in the network architecture 500. The user may inherit user profile information from user profile data structures such as templates, other user access profiles, the user's group or class, or other networks that the user is set up in. A user profile data structure is any user profile information to be retrieved when inheriting a user access profile.

In a prior solution, a command interpreter in a single computer operating system shell environment allows a user to customize the interpretation of command strings. The user may inherit portions of other user's shell environment by adding the other user's shell attributes. User access profiles are typically stored in the access provider's database. FIGS.

5 and FIG. 8 disclose one embodiment for inheriting user access profiles in an example of the invention. The user access profile is created through an inheritance process where the user is able to select capabilities, macros, functions, methods, and data to inherit from other profiles. In this embodiment, users inherit user access profiles from classes, groups, or provider recommendations. Features of user access profiles such as alias translation could then be implemented with new users rapidly. The inheritance of user access profiles also simplifies the configuration of users. In one example, a user initiates the user access profile inheritance by clicking a button on a website. Also, when a user requests a new service, the service provider automatically inherits the user access profile for the user so the user is able to use the requested service.

FIG. 8 depicts a flowchart for the access server 524 for inheriting a user access profile in an example of the invention. FIG. 8 begins in step 800. In step 802, the access server 524 waits for the next packet. The access server 524 then receives and processes the packet from the network device 562 in step 804. The access server 524 then checks if the destination is the user access profile in step 806. If the destination is not the user access profile, the access server 524 transmits the packet on the correct path in step 808 before returning to step 802. If the destination is the user access profile, the access server 524 then checks if the network device 562 is allowed access to the user access profile in step 1110. If the network device 512 is not allowed access to the user access profile, the access server 524 discards the packet and registers a network request security event in step 812 before returning to step 802.

If the network device 512 is allowed access to the user access profile, the access server 524 then checks if the user is allowed to update their user access profile in step 814. If the user is not allowed updates to their user access profile, the access server 524 generates and transmits a profile update not allowed message to the network device 512 in step 816 before returning to step 802. If the user is allowed updates to their user access profile, the access server 524 generates and transmits a user access profile update permission message asking if the user wishes to update the user access profile to the network device 512 in step 818. The access server 524 then checks if the user approved the user access profile update in step 820. If the user does not approve, the access server 524 generates and transmits a user access profile aborted message to the network device 512 in step 822 before returning to step 802.

If the user approves the user access profile update, the access server 524 sets an external request timer in step 824. The access server 524 then generates and transmits a user access profile update request to the database system 522 in step 826. A user access profile update request could be any signaling, message, or indication to update the user access profile. The access server 524 then checks if a reply was received or the external request timer expired in step 828. If the reply was not received and the external request timer did not expire, the access server 524 returns to step 828. If the reply was received or the external request timer did expire, the access server 524 checks if the reply was valid in step 830. If the reply was valid, the access server 524 generates an update complete message in step 832 before returning to step 802. If the reply was not valid, the access server 524 discards the packet and replies to the network device 512 that the user access profile update failed in step 834 before returning to step 802.

FIG. 9 depicts a flow chart for the database system 522 for inheriting a user access profile in an example of the inven-

tion. FIG. 9 begins in step 900. In step 902, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 904. In step 906, the database system 522 then checks if the access server requester is known. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 912. The database system 522 appends the reply to the packet and transmits the reply to the access server 524 in step 910. The database system 522 then replies with a decline message to the access server 524 in step 912 before returning to step 902.

If the requester is known, the database system 522 then checks whether the request is a user access profile update request in step 914. If the request is not a user access profile update request, the database system 522 checks if the request is a security event in step 916. If the request is not a security event, the database system 522 then registers an unknown action event in step 918 before returning to step 910. If the request is a security event, the database system 522 increments a path/device security record in step 924. The database system 522 then logs the path/device security record in step 926 before returning to step 902.

If the request is a user access profile update request, the database system 522 identifies and authenticates the user and the network device 512 in step 920. The database system 522 then checks if the access is valid in step 922. If the access is invalid, the database system 522 returns to step 924. If the access is valid, the database system 522 retrieves the user access profile information from a user profile data structure in step 928. The retrieval of user access profile information may be based on any information in the user access profile such as group or class or selections of inheritance user access profile presented to the user. The group or classes could be any logical grouping of user based on similar interests or situations such as work, family, and geographic location. The database system 522 then updates the user access profile in step 930 based on the user access profile information retrieved in step 928 and the user's selections for updating. The database system 522 then replies with an approve message to the access server 524 in step 932 before returning to step 902. In one embodiment of the invention, the database system 522 uses the user access profile system 572 to perform the operations disclosed in FIG. 9 and stores the user access profile in the user access profile system 572.

45 Network Shell

The user access profile includes data to provide a customized network shell to the user. The network shell is a user interface to the access network that is linked to programs, methods, macros, or service. Typically, configuration of the network shell is graphically presented to the user on a display. For example, the network shell appears as a list of alias selections that are associated with actions such as a program or macro for resources or services. An alias selection is any information that is associated with an action to be executed when the alias selection is selected. In another example, the alias selections are graphically presented as icons that relate to actions to be executed. The network shell overlays the standard DNS offered in IP networks, which reduces alias translation delays and required user keystrokes. In prior solutions, a command interpreter in an operating system shell environment allows a user to customize the interpretation of command strings, or a DNS translates "alias" values to addresses. FIGS. 5, 10, and 11 show one embodiment for network shells in an example of the invention.

FIG. 10 depicts a flowchart of an access server for executing a network shell in an example of the invention.

13

FIG. 10 begins in step 1000. In step 1002, the access server 524 waits for the next packet. The user selects an alias selection from the network shell graphically presented. In another embodiment, the user enters an alias value by a non-graphical means. Alternatively, the network shell is not presented to the user. The network device 512 transmits a packet including the alias selection to the access server 524. In step 1004, the access server 524 receives and processes the packet from the network device 512 and processes the packet. The access server 524 then checks if the network device 512 is allowed access to the network architecture 500. If the network device 512 is not allowed access to the network architecture 500, the access server 524 discards the packet and registers a network security event in step 1008 before returning to step 1002.

If the network device 512 is allowed access to the network architecture 500, the access server 524 checks if the user is recognized in step 1010. If the user is not recognized, the access server 524 returns to step 1008. If the user is recognized, the access server 524 retrieves the user's network shell in step 1012. In some embodiments, the user's network shell is retrieved from the user access profile in the access database 522 prior to presenting the network shell to the user. In step 1014, the access server 524 checks if the packet from the network device 512 includes an alias selection from the user's network shell. In one embodiment, specialized hardware is used to scan for aliases just as IP addresses are currently scanned for. If the packet does not include an alias selection from the user's network shell, the access server 524 proceeds to step 1018. If the packet does include the alias selection from the user's network shell, the access server 524 executes the action associated with the alias selection in step 1016 before returning to step 1002. In step 1018, the access server 524 processes the packet with the normal handling before returning to step 1002.

FIG. 11 depicts a flowchart of a database system for updating a network shell in an example of the invention. FIG. 11 begins in step 1120. In step 1122, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 1124. The database system 522 then checks if the access server requester is known in step 1126. If the access server requester is not known, the database system 522 registers an unknown requester event in step 1128. The database system 522 then appends the reply to the packet and transmits the packet to the access server 524 in step 1130. In step 1132, the database system 522 replies with a decline message to the access server 524 before returning to step 1122.

If the access server requester is known, the database system 522 checks if the request is a profile update for the network shell in step 1134. If the request is a profile update for the network shell, the database system 522 identifies and authenticates the user and the network device 512 in step 1136. The database system 522 then checks whether the access is valid in step 1138. If the access is invalid, the database system proceeds to step 1154. If the access is valid, the database system 522 retrieves the user access profile in step 1140. The database system 522 then updates the user access profile with the network shell with the user's alias selections and the associated programs, macros, functions or methods in step 1142. The database system 522 then replies with an approve message in step 1144 before returning to step 1122.

If the request is not a profile update, the database system 522 checks if the request is an action event in step 1146. If the request is an action event, the database system 522

14

performs the action and replies in step 1148 before returning to step 1122. If the request is not an action event, the database system 522 checks if the request is a security event in step 1150. If the request is not a security event, the database system 522 registers an unknown request type event in step 1152 before returning to step 1122. If the request is a security event, the database system 522 proceeds to step 1154. In step 1154, the database system increments a path/device security record. The database system 522 then appends the requester information to the packet and logs the event before returning to step 1122. In one embodiment of the invention, the database system 522 uses the user access profile system 572 to perform the operations disclosed in FIG. 11 and stores the user access profile in the user access profile system 572.

Service Based Directory

In prior systems, the RADIUS server provided the user with selections for transport modes. However, the user was not able to select available network based services. FIGS. 5 and 12 disclose one embodiment for a service based directory in an example of the invention. In this embodiment, access providers provide a list of services that the user can select. With the list of services, the access providers have the ability to advertise specific services to users. The list of services may be generated based on the user access profile to make the list user specific. Once the user makes the selection, the access server 524 connects the user to the service network such as Intranet, Internet, or private dedicated network that provides the selected service.

FIG. 12 depicts a flowchart for the services based directory in an example of the invention. FIG. 12 begins in step 1200. In step 1202, the access server 524 waits for the next connection from a network device in the user network 510. The access server 524 then receives a connection from the network device 512 in step 1204. Once the connection is established, the access server 524 generates and transmits a user ID query to the network device 512 in step 1206. The access server 524 then receives an ID reply and establishes a network device session in step 1208.

The access server 524 then generates an available services reply including a list of services in 1210. In one embodiment, the access server 524 generates the available services reply based upon information in the user access profile. The access server 524 receives a selected service reply from the network device 512 in step 1212. The access server 524 then connects the network device 512 to the selected service provider in step 1214. The access server 524 waits for the next packet in step 1214. The access server 524 then exchanges packets between the network device 524 and the selected service provider in step 1218.

Access Network User Binding—FIGS. 5, 13, 14, 15, and 16

User Access Profile Mobility

Users may access their user access profile from any network device connected to the network architecture 500. In a prior solution, a command interpreter in an operating system shell environment allows a user to customize the interpretation of command strings. In this environment, the access network identifies the user and the terminal device interface, which provides user mobility. The access network then executes customized actions for the user. FIGS. 5 and 13 show one embodiment of the invention for user access profile mobility. This embodiment provides user mobility in a distributed data network environment.

FIG. 13 depicts a flowchart of user access profile mobility in an example of the invention. FIG. 13 begins at step 1300. A user at the network device 512 signs on the access network

520 with their user ID. The network device 512 transmits user information with the user ID to the access server 524. In this embodiment, the user information is in the form of a packet. In step 1302, the access server 524 receives and processes the packet to check if the user access profile is local within the local database system 570. A user access profile is local within the local database system 570 when the user access profile is located in the local database system 570. If the user access profile is local, the access server 524 proceeds to step 1312.

If the user access profile is not local within the local database system 570, the access server 524 checks if the user ID is delimited with a provider ID in step 1304. One example of a user ID delimited with a provider ID includes a user's name and a provider ID separated by a delimiter such as joesmith@access.net. If the user ID is not delimited, the access server 524 retrieves the location of the default user access profile system using a default Lightweight Directory Access Protocol (LDAP) interface system 571 in step 1308 before proceeding to step 1312.

If the user ID is delimited, the access server 524 checks if the provider ID is valid in step 1306. If the provider ID is not valid, the access server 524 registers an invalid profile request security event before terminating at step 1326. If the provider ID is valid, the access server 524 uses the provider ID to retrieve the location of the local database system 590 from a foreign LDAP interface system 591 before proceeding to step 1312.

In step 1312, the access server 524 checks if the packet is a retrieve request for the user access profile. If the packet is a retrieve request, the access server 524 generates and transmits a request to retrieve the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then creates the user session ID and updates the user entries of the user access profile with the user ID and date/time updated in step 1312. The access server 524 then transmits a reply message with the session ID and the location of the LDAP server in step 1314 before terminating at step 1326.

If the packet is not a retrieve request, the access server 524 checks if the packet is a release request in step 1316. If the packet is a release request, the access server 524 retrieves the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then updates the user session ID and updates the user entries of the user access profile with the user ID and date/time updated in step 1318. The access server 524 then transmits a reply message with the session ID in step 1320 before terminating at step 1326.

If the packet is not a release request, the access server 524 checks if the packet is an update request in step 1322. If the packet is not an update request, the access server 524 registers an invalid profile request security event before terminating at step 1326. If the packet is an update request, the access server 524 retrieves the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then updates the user access profile with the information in the packet in step 1324. The access server 524 then transmits a complete message to the user network 510 to signify the profile update is complete before terminating at step 1326. In one embodiment, the local database system 570 uses the user access profile system 572 to retrieve and store the local user access profiles and the local database system 590 uses the user access

profile system 592 to retrieve and store the foreign user access profiles.

User, Device, and Service Sessions

Access network providers provide user and device sessions in addition to service sessions to distinguish users when providing communication services. A user session is the information associated with a user accessing a network. A device session is the information associated with a device being used to access a network. A service session is the information associated with a service being provided over a network. Service providers distinguish users instead of access devices or links. This allows multiple users to share a single access device with each user receiving their own customized or preferred services. Advantageously, service providers establish service access rights and restrictions such as preventing adult content for younger viewers or sharing an access device for business and personal use. Also, user, device, and service sessions allow a service provider to group multiple service providers to provide a composite of services to the user similar to a contractor/sub-contractor relationship.

FIGS. 5, 14, 15, and 16 disclose one embodiment for device, user, and service sessions in an example of the invention. FIG. 14 depicts a logical view of device, user, and service sessions in an example of the invention. Devices 1402, 1404, and 1406 are comprised of session type specific information and session links. Session type specific information include public keys, private keys, and session ID. Session links include user sessions, device sessions, and service sessions. The public keys and private keys are for encryption and decryption of messages. Session ID identifies the session ID for the device. User sessions are user sessions that the device is logically linked to. Service sessions are the service sessions the device is logically linked to. Users 1408, 1410, and 1412 are comprised of public keys, private keys, session ID, device sessions, and service sessions. Session ID identifies the session ID for the user. Device sessions are the device sessions the user is logically linked to. Service sessions are the service sessions the user is logically linked to. Services 1414, 1416, and 1418 are comprised of public keys, private keys, session ID, user sessions, device sessions, and sub service sessions. Session ID identifies the session ID for the service. User sessions are the user sessions the service is logically linked to. Device sessions are the device sessions the service is logically linked to. Sub service sessions are the service sessions the service is logically linked to.

In one example, device 1402 is linked to user B 1410 via a link 1422. User B is also linked to service N 1418 via a link 1424. Device 1402 contains the user information in the user session fields for user B 1410 and the service information in the service session fields for service N. User B 1410 contains the device information in the device session fields for device 1402 and the service information in the service session fields for service N 1418. Service N 1418 contains the device information in the device session fields for device 1402 and the user information in the user session fields for user B 1410. Service N 1418 is also linked to service A 1414 via a link 1426. Service N 1418 contains additional service information in the service session fields for service A 1414. Service A 1414 contains the composite service information in the service session fields for service N 1418. Service A 1414 also includes an owning service which is a reference to the service that owns service A.

FIG. 15 depicts a flowchart for a user based session in an example of the invention. FIG. 15 begins in step 1500. In step 1502, the network device 512 establishes a connection

17

with the access server 524. The access server 524 generates and transmits a user ID query to the user network 510 in step 1504. In step 1506, the network device 512 receives the user ID query and transmits a user ID reply to the access server 524. The access server 524 receives and processes the user ID reply to generate a device session. The network device 512 then transmits a packet to the access server 524. The access server 524 then receives the packet in step 1508. In step 1510, the access server 524 processes the packet to check if the packet includes a user session ID.

If the packet does not include the user session ID, the access server 524 transmits a login query encrypted by the network device's 512 public encryption key 1402 to the network device 512 in step 1512. The network device 512 decodes the login query with its public encryption key and transmits a login reply encrypted with its public encryption key in step 1516. The access server 524 then decodes the login reply with the private encryption key and checks if the user ID is valid. If the user is valid, the access server 524 generates a user session and a session ID in step 1520. In some embodiments, the user session ID is the original IP address. The access server 524 then encrypts with the public encryption key and transmits a complete reply with the user session ID to the network device 512 in step 1522 before returning to step 1508.

If the packet does include the user session ID, the access server 524 checks if the user and user session ID are valid in step 1514. If the user or user session ID is not valid, the access server 524 discards the packet and registers a security event in step 1515 before returning to step 1508. If the user and user session ID are valid, the access server 524 generates a service request with the user session ID and the service session ID if available. The access server 524 encrypts the service request with the public encryption key where appropriate. The access server 524 transmits the packet including the service request in step 1518 before returning to step 1508.

FIG. 16 depicts a flowchart for a service based session in an example of the invention. FIG. 16 begins in step 1600. The network device 512 transmits a service request to the access server 524. The access server 524 then receives the service request in step 1602. In step 1604, the access server 524 checks if the service request includes a service session ID.

If the service request does not include the service session ID, the access server 524 transmits a service ID query encrypted with the public encryption key to the network device 512 in step 1606. The network device 512 decodes the service ID query with its private key and transmits a service reply encrypted with the service public encryption key to the access server 524 in step 1610. The access server 524 then decodes the service reply with the service private encryption key and checks if the user is valid. If the user is valid, the access server 524 generates a service session and a session ID in step 1614. The access server 524 then transmits a complete reply with the service session ID to the network device 512 in step 1616 before returning to step 1602.

If the packet does include the service session ID, the access server 524 checks if the user and service session ID are valid in step 1608. In one embodiment, the service session ID is the destination IP address. If the user or service session ID is not valid, the access server 524 discards the packet and registers a security event before returning to step 1602. If the user and service session ID is valid, the access server 524 updates the service request with the user session ID and the service session ID. The access server 524

18

encrypts the service request with the service public encryption key where appropriate. The access server 524 transmits the service request with the user session ID and the service session ID in step 1612 before returning to step 1602.

Access Network Security—FIGS. 5, 17, and 18 Service Capability Firewall

A network service provider interfaces with a network access provider at a transport functional level. The interface typically includes Internet protocol firewalls to provide security. Unfortunately, the interface between the network service provider and the network access provider is not able to hide the implementation details such as addressing schemes, transport details, and equipment specifics. The hiding of implementation details is preferred to prevent hackers from manipulating the implementation details. FIGS. 5, 17 and 18 depict one embodiment for a service capability firewall in an example of the invention. In this embodiment, the interface between the network service provider and the network access provider occurs at a named functional level instead of the transport addressing level. A named function request is any request for a capability of service provided by the network service provider. This allows the network service provider to hide the implementation details. The network service provider exposes only functional capabilities to the users depending on their security rights.

FIG. 17 depicts a flowchart for the firewall 556 for service capability firewall in an example of the invention. FIG. 17 begins in step 1700. In step 1702, the firewall 556 waits for information. In this embodiment, the information is in the form of a packet. The firewall 556 receives and processes a packet including a named function request from the network device 512 in step 1704. The firewall 556 then checks whether the firewall 556 is the destination for the named function request in step 1706. If the firewall 556 is not the destination, the firewall 556 registers a network request security event in step 1710. In step 1712, the firewall 556 encapsulates the packet with path information and transmits the packet to the database system 522 before returning to step 1702.

If the firewall 556 is the destination, the firewall 556 checks whether the sending address is consistent with the path in step 1714. If the sending address is not consistent with the path, the firewall 556 returns to step 1710. If the sending address is consistent with the path and does not belong to the accessing network, the firewall 556 checks if the sending address/session ID/named function combination is cached in step 1716. If the sending address/session ID/named function combination is cached, the firewall 556 replaces the packet's named function request with the private cached destination address in step 1718. The firewall 556 then checks if the private destination address is known and allowed to pass in step 1720. If the destination is known and allowed to pass, the firewall 556 transmits the packet on the correct path with standard firewall access in step 1722 before returning to step 3802. If the destination is not known or not allowed to pass, the firewall 556 returns to step 1710.

If the sending address/session ID/named function combination is not cached, the firewall 556 set an external request timer in step 1724. The firewall 556 then generates and transmits a request for the database system 522 in step 1726. The firewall 556 checks whether a reply is received or the timer has expired in step 1728. If the reply is not received and the timer has not expired, the firewall 556 returns to step 1728. In another embodiment, a blocked I/O is used instead of the wait loop in step 1728. If the reply is received or the timer has expired, the firewall 556 checks if there is a valid

reply in step 1730. If the reply is invalid or no reply was received, the firewall 556 discards the packet and registers a translation failure event in step 1734 before returning to step 1702. If the reply is valid, the firewall 556 replaces the packet's named function request based on the reply and caches the address/session functions returned in step 1732 before returning to step 1720.

FIG. 18 depicts a flowchart for the database system 522 for service capability firewall in an example of the invention. FIG. 18 begins in step 1800. In step 1802, the database system 522 waits for the next request. The database system 522 then receives and processes the request in step 1804. In step 1806, the database system 522 then checks if the access server requester is known. If the access server requester is not known, the database system 522 registers an unknown requester event in step 1808. The database system 522 then appends the reply with the packet and replies to the access server 524 in step 1810. The database system 522 then generates and transmits a decline reply to the access server 524 in step 1812 before returning to step 1802.

If the requester is known, the database system 522 then checks if the request is a capability appeal in step 1814. A capability appeal is any message, signaling or instruction requesting a capability with a related network address in the service provider. If the request is not a capability appeal, the database system 522 then checks if the request is a security event in step 1816. If the request is not a security event, the database system 522 register an unknown request event in step 1817 before returning to step 1810. If the request is a security event, the database system 522 increments a path/device security record and appends the requester information to the request packet and logs the event in step 1822 before returning to step 1802. If the request is a capability appeal, the database system 522 then identifies the user and network device 512 in step 1818. The database system 522 then checks if the access is valid in step 1822. If the access is invalid, the database system 522 returns to step 1822. If the access is valid, the database system 522 retrieves the user and network device profiles in step 1824. The database system 522 then checks whether the capability is valid for the user and session state in step 1826. If the capability is invalid, the database system 522 returns to step 1822. If the capability is valid, the database system 522 generates a session ID, the network address of the capability requested, and functions available and appends the network address, which is only sent to the firewall/router 526 and not to the user, to the reply in step 1830. The database system 522 then transmits the approve reply to the access server 524 in step 1832 before returning to step 1802.

Access Network Service Authorization—

FIGS. 5, 19, 20, 21, 22, 23, and 24

Prepaid Access

Prepaid phone cards are commonly used in PSTN, where the customer pays a prepaid amount that is debited against when the customer makes a call. In data networks, prepaid cards are not currently being used. FIGS. 5 and 19 disclose one embodiment for prepaid access in an example of the invention. In this embodiment, users buy prepaid cards from network providers. When the user requests access to one of many access providers throughout the country, the access provider verifies the prepaid account code before providing the access. A prepaid account code is any number that relates to a user's prepaid account. The prepaid account is debited against for the charges related to the access. Other charges related to the service provided may also be debited against the prepaid account. For example, a user may purchase an item from a website and have the charges debited against the

prepaid account. Once the prepaid amount is reached, the access provider terminates the access to the user. In one embodiment, the network provider provides different levels of service such as gold, silver, and bronze. The gold service has guaranteed throughput but higher rates for access, while the bronze service has lower throughput and rates.

FIG. 19 depicts a flowchart for prepaid access in an example of the invention. FIG. 19 begins in step 1900. In step 1902, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 1904. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits information identifying a billing code. A billing code is any number that identifies a user for billing. Two examples of billing codes are prepaid account codes and credit card numbers. In this embodiment, the information identifying a billing code is a response including a prepaid account code to the login query. In other embodiments, the billing code is retrieved from the user access profile. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response including the prepaid account code to determine if the user is known in the local database system 570 in step 1906. A user is known when the user is allowed to use the access network 520. In one embodiment, the user is known in the local database system 570 if there is time/amounts left in the user's prepaid account. In another embodiment, the database system 522 evaluates a positive balance file to determine the remaining time/amount in the user's prepaid account. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 1908 before returning to step 1902.

If the user is not known, the database system 522 checks if there is an appeal server for user authentication in step 1910. In one embodiment, the access server 524 performs the appeal on a decline. If there is no appeal server, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2412 before returning to step 2402. If there is an appeal server, the database system 522 generates an authorization query including the prepaid account code for the central database system 580 in step 1914. The database system 522 checks if the user is known in the central database system 580 in step 1914. In one embodiment, the user is known in the central database system 580 if there is time/amounts left in the user's prepaid account. If the user is known, the database system 522 proceeds to step 1908. If the user is not known, the database system 522 proceeds to step 1910.

In one embodiment, the database system 522 uses a user authorization system 575 for checking if the user is known in the local database system 570. The user authorization system 575 contains all the prepaid customers, prepaid customer information, prepaid account codes, and the amount/quantity remaining in the prepaid account to verify if access is allowed. In one embodiment, the database system 522 uses a user authorization system 581 as the appeal server for checking if the user is known in the local database system 580. The user authorization system 581 contains all the prepaid customers, prepaid customer information, and the amount/quantity remaining in the prepaid account.

Bank Card Access

In prior systems, access providers authenticated users using their own databases. FIGS. 5, 20 and 21 disclose one

embodiment for bank card access in an example of the invention. In this embodiment, access providers authenticate users through bank card financial networks using the users' credit card numbers. Network providers use credit or debit card numbers as user ID and passwords for authentication and authorization purposes. Users use prepaid cards, phone cards, and credit cards in PSTN. However, no bank cards have been used for access to data networks other than for batch bill payment.

FIG. 20 depicts a flowchart for bank card access for a connection in an example of the invention. FIG. 20 begins in step 2000. In step 2002, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2004. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits information identifying a billing code. In this embodiment, the information identifying a billing code is a response including credit card numbers to the login query. In other embodiments, the billing code is retrieved from the user access profile. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response to see if the user is known in the local database system 570 in step 2006. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 2008 before returning to step 2002.

If the user is not known, the database system 522 processes the response to check if the user is identified by credit card numbers in step 2010. If the user is not identified by the credit card numbers, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2012 before returning to step 2002. If the user is identified by the credit card numbers, the database system 522 generates an authorization query as a pre-authorization hold or authorization/capture transaction for the central database system 580 in step 2014. In one embodiment, the central database system 580 uses a financial interface 582 to interface with a financial switch to banks for authentication and authorization. The database system 522 then checks if the user is authenticated and authorized by the central database system 580 in step 2016. If the user is authenticated and authorized, the database system 522 logs access information and an authorizing financial entity or institution in step 2018 before proceeding to step 2008. If the user is not known, the database system 522 checks if there is another database system for authorization such as for foreign user access in step 2018. If there is another database system, the database system 522 returns to step 2014. If there is not another database system, the database system 522 returns to step 2012.

FIG. 21 depicts a flowchart for bank card access for a disconnection in an example of the invention. FIG. 21 begins in step 2100. In step 2102, the access server 524 waits for a disconnection. The network device 512 then disconnects from the access server 524 in step 2104. The access server 524 then checks whether the user was on a pre-authorization hold in step 2106. If the user is not on a pre-authorization hold, the 522 logs access information and an authorizing database system 522 in step 2108 before proceeding to step 2102. If the user is on a pre-authorization hold, the database system 522 generates a pre-authorization complete transaction for the central database system 580 in step 2110 before returning to step 2108.

Access Cards

In prior systems, access providers authenticated users using their own databases. FIG. 5, 22 and 23 disclose one embodiment for network access cards in an example of the invention. An access card is a card that a user uses to access a network. The access card includes an access card account code. The access card account code is any number that relates to the user's access account. In this embodiment, access providers authenticate users who have access cards using other access providers' databases. Access providers then may enter into sharing agreements with other access providers. These sharing agreements allow users to use multiple access providers, which provide the users greater mobility and availability. Users use phone cards in PSTN for phone card access. However, no access cards for data networks have been used.

FIG. 22 depicts a flowchart for network access cards for a connection in an example of the invention. FIG. 22 begins in step 2200. In step 2202, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2204. In another embodiment of the invention, the user calls a toll free number to request access. A service control point is queried to determine where to route the request for access similar to an automatic call distribution (ACD). The request for access is then routed to the access server 524 to establish a connection between the network device 512 and the access server 524.

Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits a response including the access card account code to the login query. The access server 524 then transfers the response from the network device 512 to the database system 522. In this embodiment, the database system 522 checks if the user is allowed to use the access network 520 by checking whether the user is known. A user is known when the user is allowed to use the access network 520. For example, a user is known when their user access profile is located within the local database system 570 and their user access profile allows the user to use the access network 520. The database system 522 receives and processes the response including the access card account code to check if the user is known in the local database system 570 in step 2206. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 2208 before returning to step 2202.

If the user is not known, the database system 522 checks if the response included foreign network account information in step 2210. Foreign network account information is any information that is indicative of an account that is external to local database system 570 that the user is attempting to gain access. If there is no foreign network account information, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2212 before returning to step 2202.

If there is foreign network account information, the database system 522 identifies the local database system 590 based on the foreign network account information and generates an authorization query for the local database system 590 in step 2214. The database system 522 then checks if the user is authenticated and authorized by the local database system 590 in step 2216. If the user is authenticated and authorized, the database system 522 logs contract and settlements information returned by the local database system 590 or indicated by the database system 522 in relation to local database system 590 in step 2218 before proceeding to step 2208. If the user is not known, the

23

database system 522 proceeds to step 2212. In one embodiment, the local database system 570 uses the user authorization system 575 to check if the user is known in the local database system 570. In one embodiment, the local database system 590 uses the user authentication system 593 for authentication and authorization.

FIG. 23 depicts a flowchart for network access cards for a disconnection in an example of the invention. FIG. 23 begins in step 2300. In step 2302, the access server 524 waits for a disconnection. The network device 512 then disconnects from the access server 524 in step 2304. The access server 524 then generates and transmits a logoff query for the database system 522. The database system 522 then transfers the logoff query to the local database system 570 and the local database system 590. The database system 522 logs the access information and the authorizing database system in step 2308 before returning to step 2302.

Global Authentication

In prior systems, access providers authenticated users using their own databases. FIGS. 5 and 23 disclose one embodiment for global authentication in an example of the invention. In this embodiment, access providers authenticate users using a centralized database. Access providers then may enter into sharing agreements with other access providers. These sharing agreements allow users to use multiple access providers, which provide the users greater mobility. Currently, cellular telephone companies enter into these sharing arrangements for greater coverage. However, this sharing of databases for authentication and authorization has not occurred for data networks.

FIG. 24 depicts a flowchart for global access in an example of the invention. FIG. 24 begins in step 2400. In step 2402, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2404. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits a response to the login query. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response to see if the user is native in the local database system 570 in step 2406. In this embodiment, the database system 522 checks if the user is allowed to use the access network 520 by checking whether the user is native. A user is native when the user is allowed to use the access network 520. For example, a user is native when their user access profile is located within the local database system 570 and their user access profile allows the user to use the access network 520. If the user is native, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 4308 before returning to step 2402.

If the user is not native, the database system 522 checks if there is an authentication/authorization server in the database system 522 for a foreign network for user authentication in step 1910. If there is no authentication/authorization server in the database system 522 for the foreign network, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2412 before returning to step 2402. If there is an authentication/authorization server in the database system 522 for the foreign network, the database system 522 generates an authorization query for the central database system 580 in step 2414. The database system 522 then checks if the user is known in the central database system 580 in step 2414. If the user is known, the database

24

system 522 proceeds to step 2408. If the user is not known, the database system 522 proceeds to step 2410. In one embodiment, the central database system 580 uses a user authorization system 581 to check if the user is known.

Access Network Proxy/Environment—
FIGS. 5, 25, 26, 27, 28, 29, 30, and 31
User Based Proxy

A proxy is an application that represents itself as one or more network endpoints. The proxy receives a request for services from a user at a network endpoint and acts on behalf of the user in transmitting and receiving user requests and replies. There are Internet proxy agents that are user network access bind points. However, the Internet proxy agents are not user specific, and they act to protect user interests, not network. These proxy agents provide address translation and basic firewall functionality. Client focused proxies have extended to cookie collection and password handling.

FIGS. 5, 25 and 26 disclose one embodiment for user based proxies in an example of the invention. The user based proxies obtain information for the user and establishes a user specific network presence. A benefit of having a user specific network presence is that user access is handled by a process owned by a network security certificate authority that prevents Trojan horse network attacks. User based proxy provides a single control and monitor point for a user. The proxy agents provides a bind point for all user specific access such as user profile functionality, translations, security, and caching.

FIG. 25 depicts a flowchart for the access server 524 for user based proxies in an example of the invention. FIG. 25 begins in step 2500. In step 2502, the access server 524 waits for the next event. If the next event is a connection, the network device 512 connects with the access server 524 in step 2504. In step 2506, the access server 524 then collects user information and transfers the user information to the database system 522 to perform the security. In one embodiment, the database system 522 uses a security server 576 to perform the security. The access server 524 then checks if the user is allowed access in step 2508. If the user is not allowed access, the access server 524 disconnects the user in step 2510 before returning to step 2502.

If the user is allowed access, the access server 524 checks if a proxy is available in the database system 522 in step 2512. In one embodiment of the invention, the user proxies are in the host system 575. If the proxy is available, the access server 524 proceeds to step 2516. If the proxy is not available, the access server 524 starts the proxy in the database system 522 in step 2514 before proceeding to step 2516.

The access server 524 checks if the user access profile information is available in step 2516. If the user access profile information is not available, the access server 524 generates and transmits a user profile error to the network device 512 in step 2518 before returning to step 2502. If the user profile information is available, the access server 524 configures the proxy for the user in step 2520. The access server 524 then generates and transmits a message with the address of the user proxy and the public encryption key to the network device 512 in step 2522. The access server 524 also collects and transmits the access address of the network device 512 and the public encryption key to the user proxy. The access server 524 then returns to step 2502.

If the next event is a disconnection, the network device 512 disconnects with the access server in step 2524. The access server 524 then generates and transmits a reset command to the user proxy to clear the proxy state information and configuration in step 2526 before returning to

step 2502. If the next event is a status/request event from the user proxy, the access server 524 sets a user proxy reply timer in step 2528. The user proxy has written a status to the access server 524, and the user proxy receives a reply in step 2530 before returning to step 2502. If the next event is a timer expiration, the access server 524 receives the user proxy reply timer expiration in step 2532. The access server 524 then generates and transmits a continue wait message and status to the user proxy in step 2534 before returning to step 2528.

FIG. 26 depicts a flowchart for a user proxy for user based proxies in an example of the invention. FIG. 26 begins in step 2600. In step 2602, the user proxy waits for the next event. If the next event is the completion of the initialization, the user proxy checks if the initialization is complete in step 2604. The user proxy then sets a request timer in step 2606. The user proxy then request work from the access server 524 and transmits a status to the access server 524 in step 2608 before returning to step 2602. If the next event is the expiration of the request timer, the user proxy checks if the request timer expired in step 2610. The user proxy then registers a timeout in step 2612 before returning to step 2606.

If the next event is an access server 524 reply, the user proxy receives the access server 524 reply in step 2614. The user proxy then checks if the reply is a continue in step 2616. If the reply is a continue, the user proxy returns to step 2606. If the reply is not a continue, the user proxy processes the configuration information or action in step 2618 before returning to step 2606. If the next event is user request, the user proxy receives the user request for service in step 2620. The user proxy then checks if the user is valid in step 2622. If the user is valid, the user proxy then checks if the request is valid in step 2624. If the request is valid, the user proxy packages the request with the security certification and encryption in step 2626. The user proxy then transmits the request to the appropriate network destination in step 2628 before returning to step 2602. If the user or request is not valid, the user proxy registers a security event in step 2630 before returning to step 2602. The user proxy exchanges user information between the user network 510 and the appropriate network destination.

Subscriber Based Proxy

Another type of proxy is a subscriber based proxy. A subscriber is a logical entity such as an organization, corporation, or other grouping of users that has subscribed to services with a service provider. FIGS. 5, 27 and 28 disclose one embodiment for subscriber based proxies in an example of the invention. The subscriber based proxies obtain information for a user of a subscriber group and establishes a subscriber specific network presence. The benefit of having a subscriber specific network presence is that user access rights of the subscriber can be handled as a group, and group rights are owned by a network security certificate authority. The subscriber proxy agents provide a bind point for all subscriber specific access such as user profile functionality, translations, security, and caching.

FIG. 27 depicts a flowchart for the access server 524 for subscriber based proxies in an example of the invention. FIG. 27 begins in step 2700. In step 2702, the access server 524 waits for the next event. If the next event is a connection, the network device 512 connects with the access server 524 in step 2704. In step 2706, the access server 524 then collects user information and transfers the user information to the database system 522 to perform the security. In one embodiment, the database system 522 uses a security server 574 to perform the security. The access server 524

then checks if the user is allowed access in step 2708. If the user is not allowed access, the access server 524 disconnects the user in step 2710 before returning to step 2702. If the user is allowed access, the access server 524 checks if all required subscriber proxies are available in the database system 522 in step 2712. In one embodiment of the invention, the subscriber proxies are in the host system 575. If the proxies are available, the access server 524 proceeds to step 2716. If the proxies are not available, the access server 524 starts the required proxies in the database system 522 in step 2714 before proceeding to step 2716.

In step 2716, the access server 524 checks if the subscriber access profile information is available. If the subscriber access profile information is not available, the access server 524 generates and transmits a subscriber profile error to the network device 512 in step 2718 before returning to step 2702. If the subscriber profile information is available, the access server 524 configures the subscriber proxies for the subscriber in step 2720. The access server 524 then generates and transmits a message with the address of the subscriber proxy and the public encryption key to the network device 512 in step 2722. The access server 524 also collects and transmits the access address of the network device 512 and the public encryption key to the subscriber proxies. The access server 524 then returns to step 2702.

If the next event is a disconnection, the network device 512 disconnects with the access server in step 2724. The access server 524 then generates and transmits a reset command to the subscriber proxy in step 2726 before returning to step 2702. If the next event is a status/request event from the subscriber proxy, the access server 524 sets a subscriber proxy reply timer in step 2728. In step 2730, the subscriber proxy has written a status to the access server 524, and the subscriber proxy receives a reply before returning to step 2702. If the next event is a subscriber proxy timer expiration, the access server 524 receives the subscriber proxy reply timer expiration in step 2732. The access server 524 then generates and transmits a continue wait message and a status to the subscriber proxy in step 2734 before returning to step 2730.

FIG. 28 depicts a flowchart for a subscriber proxy for subscriber based proxies in an example of the invention. FIG. 28 begins in step 2800. In step 2802, the subscriber proxy waits for the next event. If the next event is the completion of the initialization, the subscriber proxy checks if the initialization is complete in step 2804. The subscriber proxy then sets a request timer in step 2806. The subscriber proxy then request work from the access server 524 and transmits a status to the access server 524 in step 2808 before returning to step 2802. If the next event is the expiration of the request timer, the subscriber proxy checks if the request timer expired in step 2810. The subscriber proxy then registers a timeout in step 2812 before returning to step 2806.

If the next event is an access server 524 reply, the subscriber proxy receives the access server 524 reply in step 2814. The subscriber proxy then checks if the reply is a continue in step 2816. If the reply is a continue, the subscriber proxy returns to step 2806. If the reply is not a continue, the subscriber proxy processes the configuration information or action in step 2818 before returning to step 2806. If the next event is a user request for service, the subscriber proxy receives the user request in step 2820. The subscriber proxy then checks if the user is valid in step 2822. If the user is valid, the subscriber proxy then checks if the request is valid in step 2824. If the request is valid, the subscriber proxy packages the request with the security

certification and encryption in step 2826. The subscriber proxy then transmits the request to the appropriate network destination in step 2828 before returning to step 2802. If the user or request is not valid, the subscriber proxy registers a security event in step 2830 before returning to step 2802. The subscriber proxy exchanges user information between the user network 510 and the appropriate network destination.

Dynamic Proxies

One prior system named "pluxy" allows a manual extension of a proxy. Pluxy requires manual determination and loading of a dynamic set of services, which is done on an operator basis. FIGS. 5 and 29 disclose one embodiment for dynamic proxies in an example of the invention. In this embodiment, both the user based proxies and the subscriber based proxies are extended in response to user actions. The dynamic proxy can be modified and enhanced to evolve with new services and/or protocols. The dynamic proxy size is smaller and more efficient by supporting only the logic that is being used. Implementation and development times of new service and protocol are also reduced because new proxies are not required with the new service and/or protocol.

FIG. 29 depicts a flowchart for a dynamic proxy for dynamic proxies in an example of the invention. FIG. 29 begins in step 2900. In step 2902, the dynamic proxy waits for the next event. If the next event is the completion of the initialization, the dynamic proxy checks if the initialization is complete in step 2904. The dynamic proxy then sets a request timer in step 2906. The dynamic proxy then request work from the access server 524 in step 2908 before returning to step 2902. If the next event is the expiration of the request timer, the dynamic proxy checks if the request timer expired in step 2910. The dynamic proxy then registers a timeout in step 2912 before returning to step 2906.

If the next event is an access server 524 reply, the dynamic proxy receives the access server 524 reply in step 2914. The dynamic proxy then checks if the reply is a continue in step 2916. If the reply is a continue, the dynamic proxy returns to step 2906. If the reply is not a continue, the dynamic proxy processes the configuration information in step 2918 before returning to step 2902. If the next event is a user request, the dynamic proxy receives the user request in step 2920. The dynamic proxy then checks if the user and the request are valid in step 2922. If the user and the request are valid, the dynamic proxy checks if there is a new service or protocol requested in step 2924. If there is a new service or protocol, the dynamic proxy proceeds to step 2934. If there is not a new service or protocol, the dynamic proxy packages the request with the security certification and encryption in step 2926. The dynamic proxy then transmits the request to the appropriate network destination in step 2928 before returning to step 2902. If the user or request is not valid, the subscriber proxy registers a security event in step 2930 before returning to step 2902.

If the next event is a new service/protocol request, the dynamic proxy receives the new service/protocol request in step 2932. In step 2934, the dynamic proxy processes the new service/protocol request to generate a handler request for a new service/protocol handler in step 2934 before returning to step 2926. The dynamic proxy receives the handler and executes the handler for the new service or proxy. In one embodiment, the dynamic proxy is enhanced by loading apple-like extensions from a handler system similar to a web browser.

Access Execution Environment

Proxy servers in the Internet represents a user in one network as existing in another network to obtain services for

the user. One problem with Internet proxy servers is the absence of the management of network resource utilization. In TINA-C, the User Agent and the User Session Manager are restricted to known service binding because any new service requires knowledge of or logic for a new CORBA interface. One problem with TINA-C is the absence of any provider resource management capabilities. Unfortunately, both Internet proxy servers and TINA-C do not provide any means of associating or viewing all network resources used by a user access session. Another problem is the lack of securing provider network resource against tampering by accessing users. Also, Internet proxy servers and TINA-C do not provide a platform-supported means of enforcing security levels on network access users. Both system also lack a validation of potential execution capability extensions.

FIGS. 5, 30, and 31 disclose one embodiment for an access execution environment in an example of the invention. The access execution environment easily and efficiently manages and secures network access by providing an execution environment in the access provider environment. Users access shared services and resources through their specific access execution environment. Because the access execution environment is in the access provider environment, the access provider can view the all of the user's activity by observing the execution environment. In this embodiment, the execution environment is setup as a standard system user on the network access platform, which allows management of the system user by operating system level user resource controls, quotas, and management mechanisms.

The access execution environment includes a multi-tasking virtual machine, a script interpreter, alias resolution capabilities, security certificate authentication, configuration handler, session caching, and protocol handling components. FIG. 30 depicts a block diagram of the access execution environment in an example of the invention. In FIG. 30, a network access platform 3000 comprises an execution environment manager 3010, a program and attribute database 3020, an execution environment A 3030, a transport A 3032, an execution environment B 3040, a transport B 3042, an execution environment C 3050, a transport C 3052, an execution environment D 3060, a transport D 3062, an execution environment E 3070, and a transport E 3072. In one embodiment, the network access platform 3000 is included within the database system 522. In another embodiment, the network access platform is included within the host system 575. The transport A 3032 is connected to the execution environment A 3030. The transport B 3042 is connected to the execution environment A 3040 and a workstation 3080. The transport C 3052 is connected to the execution environment C 3050. The transport D 3062 is connected to the execution environment D 3060. The transport E 3072 is connected to the execution environment E 3070. The execution environment A 3030 includes a transport handler 3034, a configuration handler 3036, and a security handler 3038. The execution environment B 3040 includes a transport handler 3044, a configuration handler 3046, and a security handler 3048. The execution environment C 3050 includes a transport handler 3054, a configuration handler 3056, and a security handler 3058. The execution environment D 3060 includes a transport handler 3064, a configuration handler 3066, and a security handler 3068. The execution environment E 3070 includes a transport handler 3074, a configuration handler 3076, and a security handler 3078.

In operation, a system administrator sets up a number of network access user accounts on the hardware platform. The administrator then sets up resource limits for each account/

platform resource. The administrator then starts an execution environment **3030**, **3040**, **3050**, **3060**, and **3070** for each access user account ID. The execution environment **A 3030** initializes and loads a configuration handler **3036**. The execution environment **A 3030** then loads the security handler **3038** and the transport handler **3034**. The transport handler **3034** opens a logical or physical transport port **A 3032** on the hardware platform using port information from the configuration handler.

FIG. 31 depicts a flow chart for the access execution environment in an example of the invention. FIG. 31 begins in step **3100**. In step **3102**, the security handler **3038** waits for a connect message from the transport handler **3034**. The security handler **3038** then receives the connect message from the transport handler **3034** in step **3104**. The security handler **3038** then generates and transmits a logon request via the transport handler **3034** in step **3106**. The security handler **3038** then checks if a reply for the logon request from the transport handler **3034** has been received in step **3108**. If the reply for the logon request has not been received, the security handler **3038** checks if a reply timeout has occurred in step **3110**. If a reply timeout has not occurred, the security handler **3038** returns to step **3108**. If a reply timeout has occurred, the security handler **3038** generates and transmits a disconnect message to the transport handler **3034** in step **3112** and returns to step **3102** to wait for a connect message.

In step **3114**, the security handler **3038** receives the reply for the logon request. Then the security handler **3038** retrieves the location of a security server from the configuration handler **3036** and transmits the logon information to the security server in step **3116**. The security handler **3038** then checks if a reply to the logon information from the security server has been received in step **3118**. If no reply has been received, the security handler **3038** checks if a reply timeout has occurred in step **3120**. If a reply timeout has not occurred, the security handler **3038** returns to step **3118**. If a reply timeout has occurred, the security handler **3038** checks if the logon information has been sent three times in step **3122**. The number of tries could be configurable. If the logon information has not been sent three times, the security handler **3038** returns to step **3116** to transmit the logon information. If the logon information has been sent three times, the security handler **3038** returns to step **3112** to send a disconnect message.

If the security server replied before the reply timeout, the security handler **3038** checks if the reply is an accept message in step **3124**. If the reply is a decline message, the security handler **3038** returns to step **3112**. If the reply is an accept message, the security handler **3038** transmits the accept message configuration parameters to the configuration handler **3036** in step **3126**. The configuration handler **3036** then loads attributes and programs and executes programs specified by the security server reply in step **3128**. The execution environment **A 3030** performs the execution of programs for the user in step **3130**. In another embodiment, the programs request attributes and/or programs to be loaded and/or executed. The programs include the ability to interface with users via the transport handler and load/execute other programs required by request types.

The configuration handler **3036** then checks if the transport handler **3034** receives a disconnect message in step **3132**. If the transport handler **3034** has not received a disconnect message, the configuration handler **3036** returns to step **3130**. If the transport handler **3034** has received a disconnect message, the transport handler **3034** transmits the disconnect message to the configuration handler **3036** in step

3134. In step **3136**, the configuration handler **6536** based on port configuration attributes closes the transport port **3032**, resets the transport handler **3034**, and notifies the execution environment manager **3010** to create a new execution environment for that port. The configuration handler **3036** gracefully shuts down all handlers, programs, and eventually the execution environment **A 3030**. The configuration handler **3036** transmits a shutdown message to the execution environment manager **3010** in step **3138** before terminating in step **3140**. The execution environment manager **3010** restarts the execution environment **A 3030** upon notification of the shutdown. The operations of the other execution environments **3040**, **3050**, **3060**, and **3070** perform in the same manner as the execution environment **A 3030** and are not discussed for the sake of clarity.

Access Network Translations—FIGS. 5, 32, 33, 34, 35, 36, 37, 38, and 39

Domain Name Scoping

The typing in of domain names such as www.nypostonline.com is quite cumbersome for the user to access specific services. The Domain Name Server (DNS) translates the domain name to a network address for the service. Clicking on bookmarks or favorites in browsers, which reference the domain names, does eliminate the need to type in the domain names. However, when users change network devices, these bookmarks or favorites do not follow the user. In an operating system environment, the command interpreter shell allows the user to customize interpretation of command strings. If no customization is loaded, the command interpreters interprets in a default fashion.

FIGS. 5, 32, and 33 disclose one embodiment for domain name scoping in an example of the invention. This embodiment provides a customizable network shell to overlay the standard DNS service offered in Internet Protocol based networks. Users then are able to define user specific acronyms or aliases for specific or logical services. An acronym or alias indicates domain names, macros, programs, actions, or network addresses. If the translation for the alias does not exist in the list of aliases for the user, the access server **524** checks if the alias exists in the list for a group in which the user belongs. The access server **524** then checks if the alias exist in the DNS for the general network. In another embodiment, the database system **522** checks the existence of the alias in the list for the group and in the DNS for the general network. There are numerous variations in the hierarchy for searching for an alias but are not discussed for the sake of simplicity.

FIG. 32 depicts a flowchart for the access server **524** for domain name scoping in an example of the invention. FIG. 32 begins in step **3200**. In step **3202**, the access server **524** waits for a packet. In this embodiment, the access server **524** receives information in the form of packets from the network device **512**. The access server **524** then receives and processes the packet from the network device **512** in step **3204**. The access server **524** then checks if the packet is an alias translation request in step **3206**. If the packet is not an alias translation request, the access server **524** transmits the packet with standard access logic in step **3214** before returning to step **3202**.

If the packet is an alias translation request, the access server **524** then checks if the alias translation is cached for the user in step **3208**. If the alias translation is cached, the access server **524** reformats the packet using the cached alias translation in step **3216** before returning to step **3214**. If the alias translation is not cached, the access server **524** sets an external request timer in step **3210**. The access server **524** then generates and transmits an alias translation request to

the database system 522 in step 3212. The alias translation request is any message, instruction, or signaling indicative of requesting an alias translation. The access server 524 checks whether a reply has been received or the external request timer has expired in step 3218. If a reply has not been received and the external request timer has not expired, the access server 524 returns to step 3218.

If a reply has been received or the external request timer has expired, the access server 524 checks if the reply was valid in step 3220. If the reply is invalid, the access server 524 discards the packet and replies with an invalid alias message or alias error message to the network device 512 before returning to step 3202. If the reply is valid, the access server 524 caches the alias translation for the user in step 3222 before returning to step 3216.

FIG. 33 depicts a flowchart for the database system 522 for domain name scoping in an example of the invention. FIG. 33 begins in step 3300. In step 3302, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 3304. In step 3306, the database system 522 then checks if the access server requester is known in step 3306. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 3308. The database system 522 appends the reply information to the packet in step 3310. The database system 522 replies with a decline message to the access server 524 in step 3312 before returning to step 3302.

If the access server requester is known, the database system 522 then checks whether the request is an alias translation request in step 3314. If the request is not an alias translation request, the database system 522 registers an unknown request event in step 3316 before returning to step 3310. If the request is an alias translation request, the database system 522 identifies and authenticates the user and the network device 512 in step 3318. If the access is invalid, the database system 522 increments a path/device security record in step 3322. The database system 522 then appends the requester denial information to the request packet in step 3324. The database system 522 then replies with a decline message to the access server 524 in step 3312 before returning to step 3302.

If the access is valid, the database system 522 retrieves the user's alias translation list in step 3326. The database system 522 then translates the alias for the user and appends the translation to a reply to the access server 524. The database system 522 then replies with an approve message to the access server 524 in step 3330. In one embodiment of the invention, the database system 522 uses the alias translation system 578 to perform the operations disclosed in FIG. 33 and stores the aliases and alias translations in the alias translation system 578.

Inband Domain Name Service Lookup

The Domain Name Server (DNS) translations of a domain name to a network address delays user interaction with the service provider. In prior solutions in voice telephone Signaling System #7 networks, a signal transfer point performs an inband local number portability (LNP) lookup on a ISDN part cell setup request such as an Initial Address Message (IAM). This inband lookup eliminates the network switch from launching a Transaction Capabilities Application Part (TCAP) LNP query to translate the telephone number in the IAM.

FIGS. 5 and 34 disclose one embodiment for inband domain name service in an example of the invention. This embodiment provides a cache for the access server 524 for alias translations on a per user basis. Thus, the external

queries for translations of aliases to domain names, macros, programs, or network addresses are eliminated. If the translated value of a request is in the alias translation cache, no translation request will be required to a DNS server. Therefore, users experience reduced delays when requesting a service.

FIG. 34 depicts a flowchart for the access server 524 for an inband domain name service lookup in an example of the invention. FIG. 34 begins in step 3400. In step 3402, the access server 524 waits for a packet. The access server 524 then receives and processes the packet from the network device 512 in step 3404. The access server 524 then checks if the destination network address and user is valid in step 3406. If the destination network address and user is valid, the access server 524 transmits the packet on the correct path to reach the destination in step 3408 before returning to step 3402.

If the destination network address or the user is invalid, the access server 524 then checks if the alias translation is cached for the user in step 3410. If the alias translation is cached, the access server 524 reformats the packet using the cached alias translation in step 3412 before returning to step 3402. If the alias translation is not cached, the access server 524 sets an external request timer in step 3414. The access server 524 then generates and transmits an alias translation request to the database system 522 in step 3416. The access server 524 checks whether a reply has been received or the external request timer has expired in step 3418. If a reply has been received and the external request timer has not expired, the access server 524 returns to step 3418.

If a reply has been received or the external request timer has expired, the access server 524 checks if the reply was valid in step 3420. If the reply is invalid, the access server 524 discards the packet and replies with an invalid alias message to the network device 512 in step 3424 before returning to step 3402. If the reply is valid, the access server 524 caches the alias translation for the user in step 3422 before returning to step 3412.

Inline Access Service Triggering

Typical firewalls filter packets out on a request by request basis on "what is not allowed". FIGS. 5 and 35 disclose one embodiment for inline access service triggering. In this embodiment, the access server 554 filters packets out on a "what is allowed" basis. Also, the access server 554 performs packet filtering on a service access basis.

FIG. 35 depicts a flowchart for inline access service triggering in an example of the invention. FIG. 35 begins in step 3500. In step 3502, the access server 554 waits for the next packet. In this embodiment, the access server 554 receives information in the form of packets from the network device 512. The access server 554 receives and processes a packet from the network device 512 in step 3504. In step 3506, the access server 554 checks if the protocol, sending address, and the destination address are known and allowed to pass. In one embodiment, the access server 554 checks the database system 522 for the determination made in step 3506. If the protocol, sending address, and the destination address are known and allowed to pass, the access server 554 checks if the sending address is consistent with the path in step 3508. If the sending address is not consistent with the path, the access server 554 registers a path security event in step 3510. The access server 554 then formats the packet with path information in step 3512. The access server 554 also increments the path/device security record and logs the event before returning to step 3502. If the sending address is consistent with the path, the access server 554 transmits the packet on the correct path with standard firewall access in step 3514 before returning to step 3502.

If the protocol, sending address, or the destination address are not known or not allowed to pass, the access server 554 checks if the request is a filter appeal in step 3515. If the request is not a filter appeal, the access server 554 discards the packet in step 3526 before returning to step 3502. If the request is a filter appeal, the access server 554 identifies and authenticates the user and the network device 512 in step 3516. The access server 554 then checks if the access is valid in step 3518. If the access is invalid, the access server 554 returns to step 3510. If the access is valid, the access server 554 retrieves the user access profile and network device's 512 profile in step 3520. The access server 554 then modifies the access logic for filter modification in step 3524. The access server 554 then modifies the protocol and address filters based on the request in step 4824 before returning to step 3514. In some embodiments, the access server 554 modifies the filters in conformance with the user access profile.

Access Service Triggering

Access providers sometimes need to extend their authentication logic beyond their primary access devices. No prior system in data networks extends the authentication logic beyond what is performed by the access provider's access devices. FIGS. 5, 36 and 37 disclose one embodiment for access service triggering. In this embodiment, the access server 524 triggers a request to external access control logic. Access providers extend the access and authentication logic beyond their access devices while maintaining centralized control of the authentication logic and user access profiles. Also, the access server 554 performs packet filtering on a service access basis.

FIG. 36 depicts a flowchart for the access server 524 for access service triggering in an example of the invention. FIG. 36 begins in step 3600. In step 3602, the access server 524 waits for the next packet. The access server 524 then receives and processes a packet from the network device 512 in step 3604. In step 3606, the access server 524 checks if the protocol, sending address, and the destination address are known and allowed to pass. In one embodiment, the access server 524 checks the database system 522 for the determination made in step 3606. If the protocol, sending address, and the destination address are known and allowed to pass, the access server 524 checks if the sending address is consistent with the path in step 3608. If the sending address is not consistent with the path, the access server 524 registers a path security event in step 3610. The access server 524 then formats the packet with path information in step 3612 before returning to step 3602. If the sending address is consistent with the path, the access server 524 transmits the packet on the correct path with standard firewall access in step 3614 before returning to step 3602.

If the protocol, sending address, or the destination address are not known or are not allowed to pass, the access server 524 sets an external request timer in step 3616. The access server 524 then generates and transmits a request with the path information to the database system 522 in step 3618. The access server 524 then checks if a reply has been received or the external request timer has expired in step 3620. If the reply has not been received or the external request timer has not expired, the access server 524 checks if the reply is valid in step 3622. If the reply is invalid, the access server 524 discards the packet in step 3624 before returning to step 3602. If the reply is valid, the access server 524 then modifies the protocol and address filters based on the reply in step 3624 before returning to step 3614.

FIG. 37 depicts a flowchart for the database system 522 for access service triggering in an example of the invention.

FIG. 37 begins in step 3700. In step 3702, the database system 522 waits for the next request. The database system 522 then receives and processes the request in step 3704. In step 3706, the database system 522 then checks if the access server requester is known in step 3706. If the access server requester is not known, the database system 522 registers an unknown requester event in step 3708. The database system 522 then appends the reply to the packet in step 3710. The database system 522 then generates and transmits a decline reply to the access server 524 in step 3712 before returning to step 3702.

If the access server requester is known, the database system 522 then checks if the request is a filter appeal in step 3714. If the request is not a filter appeal, the database system 522 then checks if the request is a security event in step 3716. If the request is not a security event, the database system 522 registers an unknown action event in step 3717 before returning to step 3712. If the request is a security event, the database system 522 increments a path/device security record, appends the requester information to the request packet, and logs the event in step 3718. The database system 522 then checks if the request is a security event in step 3719. If the request is not a security event, the database system 522 then returns to step 3712. If the request is a security event, the database system 522 proceeds to step 3728.

If the request is a filter appeal, the database system 522 then identifies the user and network device 512 in step 3720. The database system 522 then checks if the access is valid in step 3722. If the access is invalid, the database system 522 returns to step 3718. If the access is valid, the database system 522 retrieves the user and network device profiles in step 3724. The database system 522 then generates access logic and appends the access logic to a reply in step 3726. The database system 522 then transmits the approve reply to the access server 524 in step 3728 before returning to step 3702.

Personal URL

The Internet currently uses Uniform Resource Locator (URL) addresses such as www.yahoo.com so that the address is in more human readable form than the network address. Unfortunately, user cannot distinguish themselves by a network address because the user's network address changes when the user's access point changes. FIGS. 5, 38 and 39 disclose one embodiment for a personal URL. In this embodiment, a network user may publish their location on the network by a user alias. A user alias is any alias that relates to the network address of a user. Logically linking a user's current network address with a user alias allows a user to be located wherever the user accesses the network.

FIG. 38 depicts a flowchart for a personal URL lookup in an example of the invention. FIG. 38 begins in step 3800. In step 3802, the database system 522 waits for the next packet. The database system 522 then receives and processes the packet including the user alias. The database system 522 then checks if a user alias translation is cached for this user in step 3806. If the user alias translation is cached, the database system 522 replies with the current network address of the user in step 3808 before returning to step 3802.

If the user alias translation is not cached, the database system 522 sets an external request timer in step 3810. The database system 522 then generates and transmits a request for user alias translation logic in step 3812. In one embodiment, the database system 522 transmits the request for user alias translation logic to the database system that contains the user access profile. The database system 522

then checks whether a reply was received or the external request timer expired in step 3814. If the reply was not received and the external request timer did not expire, the database system 522 returns to step 3814. If a reply was received or the external request timer did expire, the database system 522 checks if the reply was valid in step 3816. If the reply was invalid, the database system 522 replies with an invalid name message in step 3824 before returning to step 3802.

If the reply was valid, the database system 522 checks if there is a current network address for the user in step 3818. If there is no current network address for the user, the database system 522 replies with user alias not currently in network message in step 3820 before returning to step 3802. If there is a current network address for the user, the database system 522 caches the user alias translation in step 3822 before returning to step 3808.

FIG. 39 depicts a flowchart for a personal URL update in an example of the invention. FIG. 39 begins in step 3900. In step 3902, the database system 522 wait for a request. The database system 522 then receives and processes the request to update the user alias in step 3904. In one embodiment, the database system receives the request from a database system that contains the user access profile. In another embodiment, the request comes from the access server 524. The database system 522 then checks if the requester is known in step 3906. If the requester is not known, the database system 522 registers an unknown requester event in step 3910. The database system 522 then transmits a reply with decline in step 3912 before returning to step 3902.

If the requester is known, the database system 522 then checks if the requester is allowed to update the user alias in step 3914. If the requester is not allowed, the database system 522 registers an unauthorized requester event in step 3916 before returning to step 3912. If the requester is allowed to update, the database system 522 then identifies and authenticates the user and current network address in step 3918. The database system 522 then checks if the access is valid in step 3920. If the access is invalid, the database system 522 increments the path/device security record in step 3922. The database system 522 also appends the requester information to the request and logs the event before returning to step 3912.

If the access is valid, the database system 522 then updates the user alias translation with the current network address in step 3924. The database system 522 then replies with an approve in step 3926 before returning to step 3902. In one embodiment of the invention, the database system 522 uses the personal DNS system 579 to perform the operations disclosed in FIGS. 38 and 39.

Access Network Caching—FIGS. 5, 40, 41, 42, 43, and 44

Predictive Caching

Users that are dialed into the network at a rate of 56 kbps or greater typically retrieve information at a rate of 2–4 kbps. Various equipment between the network access provider and the service provider cause this lower rate of transmission. Some examples of the equipment are network access provider equipment, service provider equipment, network backbone overloading, traffic shaping device, firewalls, and routers. One solution for compensating for the lower rate of transmission is caching. Computers typically contain a memory cache for specific application or devices. Firewalls and routers may also contain caches for data. Unfortunately, none of these caches are user specific.

FIGS. 5, 40, 41 and 42 disclose one embodiment of the invention for predictive end user caching. Predictive end

user caching advantageously improves user noticeable delays and slowdowns due to the lower transmission rate. The extension of data requests from the user network 510 beyond the access server 524 is eliminated when the data requested is cached in the database system. Also, using a predictive algorithm reduces delay by improving caching efficiency based on a user's predictable pattern.

FIG. 40 depicts a flowchart for the access server 524 for auditing in an example of the invention. FIG. 40 begins at step 4000. In step 4002, the user network 510 establishes a connection to the access server 524. In step 4004, the access server 524 generates and transmits a login request message to the database system 522. The access server 524 then checks if the database system allowed an access session to be established for the user in step 4006.

If the access session is not established, the access server 524 terminates in step 4020. If the access session is established, the access server 524 then checks if an access session tear down is requested in step 4012. If the access session tear down is requested, the access server 524 generates and transmits a tear down message with user and path information to the database system 522 in step 4014 before terminating in step 4020.

While no session tear down request is received, the access server 524 exchanges packets with the user network 510 depending on the service provided in step 4016. The access server 524 transmits all new packet destinations including the user and path information to the database system 522 in step 4018 before returning to step 4012. In one embodiment, the database system 522 stores the connect request message, the session reject message, the session accept message, the tear down message including the user and path information, and the packets including the user and path information in an audit database system 526. In one embodiment, the connect request message, the session reject message, the session accept message, the tear down message including the user and path information, and the packets including the user and path information are in the form of an audit message. An audit message is any message, information, or signaling that is audited while a user is accessing an access network 520.

FIG. 41 depicts a flowchart for the database system 522 for auditing in user predictive caching in an example of the invention. FIG. 41 begins in step 4100. In step 4102, the database system 522 waits for an audit message. In step 4104, the database system 522 receives and processes the audit message. The database system 522 then checks if the audit message is a session accept message or a tear down message in step 4106. If the audit message is not a session accept message or a tear down message, the database system 522 retrieves the session state object and updates the object with the audit message in step 4112 before returning to step 4102.

If the audit message is a session accept message or a tear down message, the database system 522 then checks if the audit message is a session accept message in step 4108. If the audit message is a session accept message, the database system 522 generates a session state object including the user, device, path, and session ID in step 4114 before returning to step 1802. If the audit message is a tear down message, the database system 522 retrieves the session state object and updates the object in step 4112. Also, the database system 522 stores the event in step 4116. In one embodiment, the database system 522 stores the event in the cache database system 574. The database system 522 removes the active reference from the session state object before returning to step 4102. In one embodiment of the invention, the database system 522 uses the audit database system 573 to perform the operations disclosed in FIG. 41.

FIG. 42 depicts a flowchart for the database system 522 for caching in user predictive caching in an example of the invention. FIG. 42 begins in step 4200. In step 4202, the database system 522 waits for the next event. The database system 522 processes the event in step 4204. In this embodiment, a request for data is a cache request. The database system 522 then checks if the event is a cache request in step 4206.

If the event is a cache request, the database system 522 identifies the user and retrieves the cached user data in step 4208. The database system 522 then checks if the requested data from the cache request is in the cached user data in step 4212. If the requested data is in the cached user data, the database system 522 replies with the requested data in step 4210 before returning to step 4202. If the requested data is not in the cached user data, the database system 522 retrieves the requested data from the service provider via the access server 524 in step 4214. The database system 524 then replies to the user network 510 with the requested data. Alternatively, in another embodiment, the access server 524 transfers the requested data from the service provider to the user network 510 and the database system 522.

In step 4222, the database system 522 updates the user access profile and the caching algorithm. The database system 522 then determines whether there is a change in the cached data based on the caching algorithm in step 4230. If there is no change in the cached data, the database system 522 returns to step 4202. If there is a change in the cached data, the database system 522 sets any new caching timers for the cached data in step 4236. The database system 522 also generates and transmits a translation change to the access server 524 to change the requested destination address to the database system 522 in step 4238 before returning to step 4202.

If the event is not a cache request, the database system 522 then checks if the event is an audit or timer request in step 4216. If the event is an audit or timer request, the database system 522 identifies the user and retrieves the user access profile in step 4218. The database system 522 then checks if the event is an audit request in step 4220. If the event is an audit request, the database system 522 proceeds to step 4222. If the event is not an audit request, the database system 522 refreshes the cached data from the service provider of the cached data based on the caching algorithm in step 4228. The database system 522 then resets the caching timer for the cached data in step 4234 before returning to step 4202.

If the event is not an audit or timer request, the database system 522 checks if the event is a reset event in step 4224. If the event is not a reset event, the database system 522 registers a unknown event received in step 4232 before returning to step 4202. If the event is a reset event, the database system 522 clears the user access profile statistics, caching timers, the user's cached data. In one embodiment, the user's cached data is stored in the cached database system 574 and the database system 522 uses the cached database system 574 to perform the operations disclosed in FIG. 42.

User Controlled Caching

Another solution to reduce user noticeable delays and slowdowns due to the lower transmission rate is user controlled caching. FIGS. 5, 40, 43 and 44 disclose one embodiment of the invention for user controlled caching. A user controls caching by creating a script of commands to cache data prior to the user accessing the network. For example, a user selects financial or local weather forecasts to cache, so when the user logs on to the network the information will be immediately available from the cache. Scripting has typi-

cally been done on general purpose computers. In a general purpose computer, the user sets up a sequenced set of commands to be executed when the script is executed. Unfortunately, this type of scripting has not been performed on a network cache.

In this embodiment of user controlled caching, the operation of the access server 524 is as described in FIG. 40. FIG. 43 depicts a flowchart for the database system 522 for auditing in user controlled caching in an example of the invention. FIG. 43 begins in step 4300. In step 4302, the database system 522 waits for an audit message. In step 4302, the database system 522 receives and processes the audit message. The database system 522 then checks if the audit message is a session accept message or a tear down message in step 4306. If the audit message is not a session accept message or a tear down message, the database system 522 retrieves the session state object and updates the object with the audit message in the cache in step 4312 before returning to step 4302.

If the audit message is a session accept message or a tear down message, the database system 522 then checks if the audit message is a session accept message in step 4308. If the audit message is a session accept message, the database system 522 generates a session state object in the cache including the user, device, path, and session ID in step 4314. The database system 522 then checks if the user has a pre-cache script in step 4318. If the user does not have a pre-cache script, the database system 522 returns to step 4302. If the user has a pre-cache script, the database system 522 generates and transmits a pre-cache instruction set to the access server 524 to change the translation filter to access the database system 522 for destinations in the pre-cache script so that any requests for those destinations are fulfilled from the cache in step 4320. The database system 522 then sets cache refresh timers in step 4322 and returns to step 4302.

If the audit message is a tear down message, the database system 522 retrieves the session state object and updates the object in step 4316. Also, the database system 522 stores the event in step 4316. In one embodiment, the database system 522 stores the session state object and the state in the cache database system 527. The database system 522 removes the active reference from the session state object before returning to step 4302. In one embodiment of the invention, the database system 522 uses the audit database system 573 to perform the operations disclosed in FIG. 43.

FIG. 44 depicts a flowchart for the database system 522 for caching in user controlled caching in an example of the invention. FIG. 44 begins in step 4400. In step 4402, the database system 522 waits for the next event. The database system 522 processes the event in step 4404. In this embodiment, a request for data is a cache request. The database system 522 then checks if the event is a cache request in step 4406.

If the event is a cache request, the database system 522 identifies the user and retrieves the cached user data in step 4408. The database system 522 then checks if the requested data from the cache request is in the cached user data in step 4412. If the requested data is in the cached user data, the database system 522 replies with the requested data in step 4410 before returning to step 4402. If the requested data is not in the cached user data, the database system 522 retrieves the requested data from the service provider via the access server 524 in step 4414. The database system 522 then replies to the user network 510 with the requested data. Alternatively, in another embodiment, the access server 524 transfers the requested data from the service provider to the user network 510 and the database system 522.

In step 4422, the database system 522 updates the user access profile and the caching algorithm. The database system 522 then determines whether there is a change in the cached data based on the caching algorithm in step 4430. If there is no change in the cached data, the database system 522 returns to step 4402. If there is a change in the cached data, the database system 522 sets the new caching timer for the cached data in step 4436. The database system 522 also generates and transmits a translation change to the access server 524 to change the requested destination address to the database system 522 in step 4440 before returning to step 4402.

If the event is not a cache request, the database system 522 then checks if the event is an audit or timer request in step 4416. If the event is an audit or timer request, the database system 522 identifies the user and retrieves the user access profile in step 4418. The database system 522 then checks if the event is an audit request in step 4420. If the event is an audit request, the database system 522 proceeds to step 4422.

If the event is not an audit request, the database system 522 checks if the event is a script timer in step 4428. If the event is not a script timer, the database system 522 refreshes the cached data from the service provider of the cached data based on the caching algorithm in step 4438 before returning to step 4402. If the event is a script timer, the database system 522 executes the user commands or command set specified by the script timer event in step 4434. The database system 522 then resets the script timer for the cached data in step 4442 before returning to step 4402.

If the event is not an audit or timer request, the database system 522 checks if the event is a reset event in step 4424. If the event is not a reset event, the database system 522 registers a unknown event received in step 4432 before returning to step 4402. If the event is a reset event, the database system 522 clears the user access profile statistics, caching timers, the user's cached data. In one embodiment, the user's cached data is stored in the cached database system 574 and the database system 522 uses the cached database system 574 to perform the operations disclosed in FIG. 44.

Access Network Switching— FIGS. 5, 45, 46, and 47

Switching Access by a User

Access between users and service providers vary based on quality and security, which in turn determine the costs of the access. Users typically have a need to switch between types of access depending on the service offered or what stage in the service the user is in. For example, a user browses the amazon.com website for books using a standard Internet access. When purchasing the books, the user needs a more secure Internet access to ensure that the user's credit card number is not stolen by a hacker.

One prior solution for enhanced security is data Virtual Private Networks (VPN). Data VPN's are relatively static constructions which allow the extension of a user network to another location by extending the network over leased lines or shared Internet/Intranet facilities. However, VPN's require service anticipation, planning and expense beyond what the typical Internet and Intranet users possess. VPN's also do not provide dynamic switching between accesses. Another prior solution is Local Area Network emulation (LANE). LANE utilizes ATM transports to extend the reach of the user network. However, most Internet and Intranet users do not possess ATM equipment and expertise, and the backbone network is still IP.

FIGS. 5 and 45 disclose one embodiment for switching access by a user in an example of the invention. A user

selects a different access to a service provider and is switched to the access without the user's connection to their access provider being interrupted. FIG. 45 depicts a flowchart for switching access by a user in an example of the invention. In this embodiment, a user using standard Internet access selects a premiere Internet access during the setup of a service application. There are numerous variations in switching between access paths such as from premiere Internet access path to a lower quality Internet access path, but only one embodiment is shown for the sake of simplicity.

FIG. 45 begins in step 4500. In step 4502, a user through the network device 512 transmits a request using a standard Internet access path through the service network 530 for a service application residing in the network device 562. The service application in the network device 562 receives the request and transmits a query for the quality of service (QOS) Internet access desired by the requester to the network device 512 in step 4504. The service application in the network device 562 then receives and processes a request to switch access to check if the user selected a premium QOS Internet access in step 4516. If the user did not select a premium QOS Internet access, the user and service application exchange packets using the standard Internet access via the network device 512, the access server 524, the service network 530, the access server 554, and the network device 562 in step 4516. The service application in the network device 562 returns the control to the user in step 4514 to select another service application in step 4502.

If the user selects a premium QOS Internet access, the service application in the network device 562 generates and transmits an authentication and authorization instruction for the premium QOS Internet access to the database system 522 to check if the switch of access is allowed. The database system 522 receives and processes the authentication and authorization instruction. The database system 522 then generates and transmits a premium access instruction to establish premium Internet access between the access server 524 and the access server 554 via the service network 540. In one embodiment, the database system 522 transmits the premium access instruction to the service network 540 to establish premium Internet access between the access server 524 and the access server 554. The database system 522 then generates and transmits the premium access instruction to the access server 524 to connect the network device 512 to the service network 540 for the premium Internet access. The database system 522 also generates and transmits the premium access instruction to the access server 554 to connect the network device 562 to the service network 540 for the premium Internet access.

Once the premium Internet access is established, the user and service application exchange packets using the premium Internet access via the network device 512, the access server 524, the service network 540, the access server 554, and the network device 562 in step 4512. The database system 522 then generates and transmits a disconnect instruction to the access server 524, the service network 540, and the access server 554 to tear down the premium Internet access in step 4518. The database system 522 then generates and transmits usage and performance statistics for the service application in the network device 562 in step 4520 before returning to step 4514. In one embodiment, the database system 522 uses a cross connect system 583 to perform the operations disclosed in FIG. 45.

Switching Access by a Service Provider

FIGS. 5 and 46 disclose one embodiment for switching access by a service provider in an example of the invention. The service provider selects a different access to a user and

41

is switched to the access without the user or service provider's connection to their access provider being interrupted. FIG. 46 depicts a flowchart for switching access by a service provider in an example of the invention. In this embodiment, the service provider selects a toll free premium Internet service for a user using standard Internet access. The service provider pays for the changes in the toll free premium Internet service instead of the user—similar to toll free phone calls.

FIG. 46 begins in step 4600. In step 4602, the service application in the network device 562 waits for a request for a service. The user through the network device 512 transmits a request using a standard Internet access through the service network 530 for the service application in the network device 562 in step 4604. The service application in the network device 562 receives and processes the request. The service application then selects a toll free premium quality of service (QOS) Internet access to the network device 512 in step 4606. The service application in the network device 562 generates and transmits an authentication instruction to the database system 522 in step 4608. The database system 522 receives and processes the authentication instruction. In step 4612, the database system 522 then generates and transmits a premium access instruction to establish the toll free premium Internet access between the access server 524 and the access server 554 via the service network 540.

Once the toll free premium Internet access is established, the user and service application exchange packets using the premium Internet access via the network device 512, the access server 524, the service network 540, the access server 554, and the network device 562 in step 4612. Once the service is completed, the database system 522 then generates and transmits a disconnect instruction to the access server 524, the service network 540, and the access server 554 to tear down the toll free premium Internet access in step 4614. The database system 522 then generates and transmits usage and performance statistics for the service application in the network device 562 in step 4616. Once the service is completed, the service application in the network device 562 returns the control to the user in step 4618 to select another service application in step 4602. In one embodiment, the database system 522 uses the cross connection system 583 to perform the operations disclosed in FIG. 46. In another embodiment, a third party such as the government performs the switching for surveillance or monitoring purposes.

Dynamic Switching Access

FIGS. 5 and 47 disclose one embodiment for dynamic switching access in an example of the invention. In a prior solution, a single access link from the user network to the access server is linked to a dedicated network in a single access session. However, users need to switch between different networks such as Internet and Intranets without tearing down the existing access session. No other systems allow the access server to be controlled by the data stream the access server is passing. Analog modems monitor the character stream for an escape sequence. The escape sequence notifies the modem that the data is for modem control and not for application data. However, no prior systems have implemented this functionality for a packet data network.

FIG. 47 depicts a flowchart for dynamic switching access in an example of the invention. In this embodiment, a user during an access session may switch networks without having the access session torn down. The access server 524 receives a control instruction from the user to switch from one dedicated service network 530 to another dedicated service network 540. FIG. 47 begins in step 4700. In step

42

4702, the access server 524 waits for a packet. In this embodiment, a request is in the format of a packet. The access server 524 then receives and processes the packet from the network device 512 in step 4704. The access server 524 checks if the packet is encoded for access control in step 4706. If the packet is not encoded for access control, the access server 524 processes the packet using standard access logic in step 4714 before returning to step 4702.

If the packet is encoded for access control, the access server 524 identifies the physical access path characteristics in step 4708. The access server 524 then checks if the access control is allowed in step 4710. If access control is not allowed, the access server 524 registers and logs an illegal access event in step 4716. The access server 524 then discards the packet in step 4718 before returning to step 4702.

If access control is allowed, the access server 524 generates and transmits an access control instruction to the database system 522 in step 4712. The database system 522 then receives and processes the access control instruction. The database system 522 identifies, authenticates, and authorizes the user and the requesting access server using the packet and path in step 4720. The database system 522 then retrieves the user access profile and the network device profile in step 4722. The database system 522 then checks if access control is allowed for the user and the network device based on the user access profile and the network device profile in step 4724. If access is not allowed, the database system 522 proceeds to step 4716.

If access is allowed, the database system 522 generates and transmits an access instruction to the access server 524 to update the access logic to switch to the service network 540 in step 4726. The database system 522 logs the access change in step 4727. In one embodiment, the database system 522 logs the access change in the audit system 572. The database system 522 also generates and transmits a reply with the new access status to the network device 512 in step 4728 before returning to step 4702. In one embodiment, the database system 522 uses the cross connect system 583 to perform the operations disclosed in FIG. 47.

Access Network Destination Control—FIGS. 5, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, and 59

Network Failover
Network devices become unavailable for a various reasons such as busy, failure, or overload. For network failover, when a destination network device is unavailable, network device requests are manually re-routed to a secondary network device. In order to avoid this manual rerouting, one prior art solution is the sharing of a network address between multiple processors in a box or multiple machines in a cluster. However, the requirement of destination and secondary devices residing in the same box or cluster makes this solution unacceptable for large systems.

Another prior art system is Network Address Translation, which translates a private network address to a public network address for a period of time. This system does not provide the capability of a network address to represent multiple failover destinations. Another solution is the Common Object Request Broker Architecture (CORBA), which provides the user a list of potential "logical" network devices to post the network device request. However, the user must implement CORBA interfaces and interactively select back up processes. The transaction context is lost when the primary fails. The user must re-establish the context. Unfortunately, none of these solutions provide an automatic translation from an unavailable destination network device to a secondary network device without operator

or user intervention, where the destination and secondary devices are not within the same box or cluster.

FIGS. 5, 48 and 49 disclose one embodiment for handling network address failover in an example of the invention. In a failover scenario, a destination network device fails or overloads, which makes the destination network device unavailable, and a secondary network device replicates the destination network device to give the user a fault tolerant appearance of the destination network device. Requests for the destination network device are translated to the secondary network device.

FIG. 48 depicts a flowchart for the access server 554 for network address failover in an example of the invention. FIG. 48 begins in step 4800. In step 4802, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 4804. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 4806, the access server 554 detects that the destination network device 562 fails and checks if the destination network address is enabled for failover in step 4806. If the destination network address is not enabled for failover, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 4812 before returning to step 4802.

If the destination network address is enabled for failover, the access server 554 checks if an address translation is active for the destination network device 562 in step 4808. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 4812 before returning to step 4802. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for failover in step 4810. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 4812 before returning to step 4802.

FIG. 49 depicts a flowchart for the database system 522 for network address failover in an example of the invention. FIG. 49 begins in step 4900. In step 4902, the database system 522 waits for a state event message or a timer event from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or a network device timer expires in step 4904. The database system 522 checks if the network device timer expired in step 4906.

If the network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout in step 4908. The database system 522 then retrieves and updates and the secondary network device's 564 profile to indicate that the destination network device 562 is unavailable in step 4910. In step 4912, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer has not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 4914. In step 4916, the database system 522 checks if the destination network device 562 is known. If the destination network device 562 is not known, the database system 522 generates

and transmits an unknown resource event before returning to step 4902. If the destination network device 562 is known, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics in step 4918. The database system 522 also resets the network device timer. The database system 522 then checks if the state event message is a failure or start event in step 4920.

If the state event message is not a failure or start event, the database system 522 checks if the load event threshold for the destination network device 562 is reached in step 4930. If the load event threshold is not reached, the database system 522 returns to step 4902. If the load event threshold is reached, the database system 522 proceeds to step 4926.

If the state event message is a failure or start event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 4922. In step 4924, the database system 522 checks if the state event message is a failure event. If the state event message is not a failure event, the database system 522 generates new active device profiles and then returns to step 4902. If the state event message is a failure event, the database system 522 proceeds to step 4926.

In step 4926, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address in step 4932 before returning to step 4902. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event in step 4928 before returning to step 4902. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 49.

Network Busy Forwarding

In a Public Switched Telephone Network (PSTN), a busy signal is sent to the caller when the called number destination is busy and thus unavailable. The caller can then select alternate actions such as delaying, retrying, or re-routing the request. Unfortunately, the PSTN handling of busy calls has not been applied to network devices in a data network. FIGS. 5, 50 and 51 disclose one embodiment for network busy forwarding in an example of the invention. When a destination network device 562 is busy, an access server 554 forwards the packets to a secondary network device 564. Requests for the destination network device are translated to the secondary network device.

FIG. 50 depicts a flowchart for the access server 554 for network busy forwarding in an example of the invention. FIG. 50 begins in step 5000. In step 5002, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5004. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5006, the access server 554 checks if the destination network device 562 is busy. If the destination network device 562 is not busy, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5012 before returning to step 5002.

If the destination network device 562 is busy, the access server 554 checks if an address translation is active for the destination network device 562 in step 5008. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step

5012 before returning to step 5002. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for failover in step 5010. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 5012 before returning to step 5002.

FIG. 51 depicts a flowchart for the database system 522 for network busy forwarding in an example of the invention. FIG. 51 begins in step 5100. In step 5102, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or the network device timer expired in step 5104. The database system 522 checks if a network device timer expired in step 5106.

If a network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout such as the network device being busy or unavailable in step 5108. The database system 522 then retrieves and updates and the secondary network device's 564 profile to indicate that the destination network device 562 is busy or unavailable in step 5110. The database system 522 also resets the network device timer. In step 5112, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer is not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 5114. In step 5116, the database system 522 checks if the destination network device 562 is known. If the destination network device 562 is not known, the database system 522 generates and transmits an unknown resource event before returning to step 5102. If the destination network device 562 is known, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics in step 5118. The database system 522 then checks if the state event message is a failure/busy or start/available event in step 5120.

If the state event message is not a failure/busy or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5130. If the load event threshold is not reached, the database system 522 returns to step 5102. If the load event threshold is reached, the database system 522 proceeds to step 5126.

If the state event message is a failure/busy or start/available event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 5122. In step 5124, the database system 522 checks if the state event message is a failure/busy event. If the state event message is not a failure/busy event, the database system 522 generates an address translation clear request in step 5132 before returning to step 5102. The address translation clear request adds a new active device profile or clears the existing translation. If the state event message is a failure/busy event, the database system 522 proceeds to step 5126.

In step 5126, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address if a busy

translation exists in the destination network device's 562 profile in step 5134 before returning to step 5102. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5128 before returning to step 5102. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 51.

Busy Flag

FIGS. 5 and 52 disclose one embodiment for a busy flag when the destination network device is busy in an example of the invention. When a destination network device 562 is busy and thus unavailable, an access server 554 replies to the user with a busy flag. FIG. 52 depicts a flowchart for the access server 554 for a busy flag when the destination network device is busy in an example of the invention. In one embodiment, the busy flag is a busy screen in HyperText Markup Language. FIG. 52 begins in step 5200. In step 5204, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5204. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5206, the access server 554 checks if the destination network device 562 is busy. If the destination network device 562 is not busy, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5210 before returning to step 5202. If the destination network device 562 is busy, the access server 554 replies to the user with a busy flag in step 5208 before returning to step 5202.

Timeout

FIGS. 5 and 53 disclose one embodiment for forwarding if the destination network device times out in an example of the invention. In this embodiment, a destination network device 562 receives a request packet from a user but fails to reply to the user within a pre-determined time. The failure to reply within a pre-determined time indicates the destination network device 562 is unavailable. The access server 554 then redirects the packet to a secondary network device 564 to handle forwarding when the destination network device fails to respond with the pre-determined time.

FIG. 53 depicts a flowchart for the access server 554 for forwarding if the destination network device times out in an example of the invention. FIG. 53 begins in step 5300. In step 5302, the access server 554 waits for the next packet or event. The access server 554 then checks whether a reply timeout event is received in step 5304. If a reply timeout event is not received, the access server 554 then receives and processes the packet in step 5306. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5308, the access server 554 checks if an address translation is available for the destination network device 562. If an address translation is not available for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5318 before returning to step 5302. If an address translation is available for the destination network device 562, the access server 554 sets a reply timer and caches the packet for the secondary network device 564. The access server 554 then transmits the packet with standard access logic to the destination network device 564 in step 5318 before returning to step 5302.

In step 5312, the access server 554 receives the reply timeout event and retrieves the cached request. The access server 554 then processes the reply timeout event and the cached request in step 5314. In step 5315, the access server

554 then replaces the packet's destination network address with the cached secondary network address. The access server 554 then transmits the packet to the secondary network device 564 with the standard access logic in step 5318 before returning to step 5302. In one embodiment, upon reply, the access server 554 deletes the cached packet copy.

Scheduled Alias Resolution

Network devices become unavailable for various reasons such as busy, failure, or overload. Also, service provider need to utilize network resources to improve efficiency and avoid network device latency. In voice telephony signaling networks, an SS7 SCP or an automatic call distributor (ACD) distributes calls to end user call centers. The SCP receives a 800 TCAP query and performs a database lookup to translate the 800 telephone number into a destination telephone number. The SCP may use the requester information and the time of day to perform the database lookup. Some problems with this system are SS7 signaling must be used and the system is limited by voice network constraints. Also, the phone numbers only allow numeric numbers for aliases. In data networks, prior art known as Object Request Brokers (ORB) in a Common Object Request Broker Architecture (CORBA) advertise process resources to a requester. The ORB also registers multiple instances of the same type of process resource.

FIGS. 5, 54, and 55 disclose one embodiment for scheduled alias resolution in an example of the invention. In this embodiment, an alias, domain name/local name, network address, or macro for a network resource is translated to another alias for another network resource based on a configurable schedule. Advantageously, load balancing of network devices is improved by scheduling different alias resolutions for different times/conditions. One problem with CORBA is the requester and resources must implement CORBA interfaces. Scheduled alias resolution is compatible with existing data network DNS implementations. Also, the load balancing, security, and failover may be based on the user access profile and the network device state.

FIG. 54 depicts a flowchart for the access server 554 for schedule alias resolution in an example of the invention. FIG. 54 begins in step 5400. In step 5402, the access server 554 waits for the next request. The access server 554 then receives and processes a request from the network device 512 in step 5404. In this embodiment, the information including an alias is in the form of a request. In step 5406, the access server 554 checks if a current translation for the alias exists. If a current translation for the alias exists, the access server 554 translates the alias to the appropriate network address for a network device in user network 560 in step 5408. The access server 554 then replies to the network device 512 in step 5410 before returning to step 5402.

If a current translation for the alias does not exist, the access server 554 checks if a known access profile exists in the database system 522 exists for this user in step 5412. If a known access profile does not exist in the database system 522 for the user, the access server 554 generates and transmits an alias resolution failure message to the network device 512 in step 5414 before returning to step 5402. If a known access profile does exist in the database system 522 for the user, the access server 554 sets an external request timer in step 5416. The access server 554 then generates and transmits an alias translation request to the database system 522 in step 5418. The access server 554 then checks if a reply is received or the external request timer is expired in step 5420. If the reply is not received or the external request timer is not expired, the access server 554 returns to step

5420. If the reply is received or the external request timer is expired, the access server 554 checks if the reply was valid in step 5422. If the reply is invalid, the access server 554 returns to step 5414. If the reply is valid, the access server 554 then caches the alias translation for this user with time to live parameters in step 5424 before returning to step 5408.

FIG. 55 depicts a flow chart for the database system 522 for scheduled alias resolution in an example of the invention. FIG. 55 begins in step 5500. In step 5502, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 554 in step 5504. In step 5506, the database system 522 then checks if the access server requester is known in step 5506. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 5508. The database system 522 appends the reply to the packet and transmits the reply to the access server 554 in step 5510. The database system 522 replies with a decline message to the access server 554 in step 5512 before returning to step 5502.

If the access server requester is known, the database system 522 then checks whether the request is a translation appeal in step 5514. If the request is not a translation appeal, the database system 522 checks if the request is a security event in step 5516. If the request is a security event, the database system 522 increments a path/device security record in step 5522 before returning to step 5512. If the request is not a security event, the database system 522 registers an unknown event in step 5530 before returning to step 5502. If the request is a translation appeal, the database system 522 identifies and authenticates the user and the network device 512 in step 5518. If the access is not valid, the database system 522 returns to step 5522. If the access is valid, the database system 522 retrieves the user and network device's 512 profiles in step 5524. The database system 522 then generates the translation logic and calculates the translation time to live in step 5526. The database system 522 then appends the translation logic and the translation time to a reply to the access server 554. The database system 522 then replies with an approve message to the access server 554 in step 5528.

Forwarding

In a PSTN, a person may forward calls from their original phone number to another phone number where the person may be reached. Unfortunately, the PSTN handling of call forwarding has not been applied to network devices in a data network. FIGS. 5, 56 and 57 disclose one embodiment for destination controlled forwarding in an example of the invention. An access server 554 forwards the packets for a destination network device 562 that is unavailable to a secondary network device 564. Requests for the destination network device 562 are translated to the secondary network device 564.

FIG. 56 depicts a flowchart for the access server 554 for destination controlled forwarding in an example of the invention. FIG. 56 begins in step 5600. In step 5602, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5604. In this embodiment, the access server 554 receives and processes information in the form of a packet. The access server 554 then checks if the packet is a reply in step 5606. If the packet is a reply, the access server 554 retrieves and deletes the cached destination network address in step 5608 before proceeding to step 5614.

If the packet is not a reply, the access server 554 checks if an address translation is active for the destination network

device 562 in step 5610. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5614 before returning to step 5602. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for destination controlled forwarding in step 5612. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 5614 before returning to step 5602.

FIG. 57 depicts a flowchart for the database system 522 for destination controlled forwarding in an example of the invention. FIG. 57 begins in step 5700. In step 5702, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or a network device timer expires in step 5704.

The database system 522 processes the state event message and authenticates the destination network device 562 in step 5706. In step 5708, the database system 522 checks if the access server requester is known. If the access server requester is not known, the database system 522 generates and transmits an unknown requester event before returning to step 5702. If the access server requester is known, the database system 522 retrieves and updates the access server's 554 profile with availability and performance statistics in step 5710. The database system 522 then checks if the state event message is a failure/forward or start/available event in step 5712.

If the state event message is not a failure/forward or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5716. If the load event threshold is not reached, the database system 522 returns to step 5702. If the load event threshold is reached, the database system 522 proceeds to step 5722.

If the state event message is a failure/forward or start/available event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 5718. In step 5720, the database system 522 checks if the state event message is a failure/forward event. If the state event message is not a failure/forward event, the database system 522 generates and transmits an address translation clear request message to the access server 554 in step 5726 before returning to step 5702. If the state event is a start event, then a new availability profile is generated. If the state event message is a failure/busy event, the database system 522 proceeds to step 5722.

In step 5722, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address if a busy translation exists in the destination network device's 562 profile in step 5728 before returning to step 5702. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmits a busy message to the user in step 5724 before returning to step 5702. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 57.

Network Endpoint Availability Management

FIGS. 5, 58, and 59 disclose one embodiment for network endpoint availability management in an example of the invention. The features of failover, busy, busy flag, timeout, and forwarding are combined into this embodiment. FIG. 58 depicts a flowchart for the access server 554 network endpoint availability management in an example of the invention. FIG. 58 begins in step 5800. In step 5802, the access server 554 waits for the next packet or event. The access server 554 then checks whether a reply timeout event is received in step 5804.

If a reply timeout event is not received, the access server 554 then receives a data packet and processes the packet in step 5806. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5808, the access server 554 checks if the packet is a reply. If the packet is a reply, the access server 554 clears the cached request and clears a reply timer in step 5810 before returning to step 5804. If the packet is not a reply, the access server 554 checks if an address translation is active for the destination network device 562 in step 5812. If an address translation is active for the destination network device 562, the access server 554 proceeds to step 5824. If an address translation is not active for the destination network device 562, the access server 554 checks if the destination address is busy. If the destination address is not busy, the access server 554 proceeds to step 5830. If the destination address is not busy, the access server 554 checks if a forward address translation is available in step 5814. If the forward address translation is not available, replies to the user with a busy flag in step 5818 before returning to step 5802. If the forward address translation is available, the access server 554 proceeds to step 5824.

If a reply timeout event is received, the access server 554 receives the reply timeout event and the cached request in step 5820. The access server 554 then processes the reply timeout event and the cached request in step 5822. In step 5824, the access server 554 then replaces the packet's destination network address with the cached secondary network address. The access server 554 then transmits the packet to the secondary network device 564 with the standard access logic in step 5824.

In step 5826, the access server 554 checks if the forward address is busy. If the forward address is busy, the access server 554 returns to step 5818. If the forward address is not busy, the access server 554 sets a reply timer and caches the request in step 5828. In step 5830, the access server 554 then transmits the packet with standard access logic before returning to step 5802.

FIG. 59 depicts a flowchart for the database system 522 for network endpoint availability management in an example of the invention. FIG. 59 begins in step 5900. In step 5902, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives the state event message from the destination network device 562 or the network device timer expired in step 5904. The database system 522 checks if the network device timer expired in step 5906.

If the network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout such as the network device being busy or unavailable in step 5908. The database system 522 then retrieves and updates the secondary network device's 564 profile to indicate that the destination network device 562 is busy or unavailable in step 5910. The database system 522 also resets the network device timer. In step 5912, the database system 522 generates and transmits an

51

address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer has not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 5914. In step 5916, the database system 522 checks if the access server requester is known. If the access server requester is not known, the database system 522 generates and transmits an unknown resource event before returning to step 5902. If the access server requester is known, the database system 522 retrieves and updates the access server's 554 profile with availability, activity logs, and performance statistics in step 5918. The database system 522 then checks if the state event message is a failure, busy, forward or start/available event in step 5920.

If the state event message is not a failure, busy, forward or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5930. If the load event threshold is not reached, the database system 522 returns to step 5102. If the load event threshold is reached, the database system 522 proceeds to step 5926.

If the state event message is a failure, busy, forward, or start/available event, the database system 522 retrieves and updates the destination network device 562 and secondary network device's 564 profile in step 5922. In step 5124, the database system 522 checks if the state event message is a failure, busy, or forward event. If the state event message is not a failure/busy event, the database system 522 generates and transmits an address translation clear request to the access server 554 if a busy translation exists in step 5934 before returning to step 5902. If the state event message is a failure, busy, or forward event, the database system 522 proceeds to step 5926.

In step 5926, the database system 522 checks if a secondary network device 564 is available for forwarding. If a secondary network device 564 is available, the database system 522 checks if the state event message is a failure or busy event in step 5936. If the state event message is a failure or busy event, the database system 522 generates and transmits a busy address translation request message to the access server 554 in step 5932 before returning to step 5902. If the state event message is not a failure or busy event, the database system 522 generates and transmits a forward address translation request message to the access server 554 to change packets' destination network address to the secondary network address in step 5938 before returning to step 5902. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5928 before returning to step 5902. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 59.

Access Network Audit Server—

FIGS. 5, 60, 61, 62, and 63

Service Capability Monitor

FIGS. 5, 60, 61, 62, and 63 disclose one embodiment for service capability monitor in an example of the invention. In a prior system, Java applet technology and Sun's JINI project delivers communication and state mechanisms from the function provider to the function requesters. Java applet technology allows a mechanism for device control without any coordinating control agent. JINI provides a registry which maintains capabilities like a CORBAORB as a means for primitive control. One problem is that JINI focuses on

52

the distribution of mechanisms instead of the interworking of mechanisms in a standardized way. In this embodiment of service capability control, a network access provider monitors the validity and state of the application level request to a network service provider. The network access provider monitors performance of their network and the service provider's network service requiring an understanding of the other provider's service requested. Access providers such as Internet service providers could mitigate their liability and increase the service's performance by extending their service view to the user base. Specific service environment access points may be monitored and service access rights and restrictions may be enforced. This embodiment provides an interworking mechanism that provides a dynamic interface to a service based network. The monitoring interface must be dynamic because different service have different potential states, different legal and illegal state transitions, and different communication mechanisms.

FIG. 60 depicts a flowchart for a firewall 526 for service capability monitor in an example of the invention. FIG. 60 begins in step 6000. In step 6002, the firewall 526 waits for a packet. The firewall 526 then receives and processes a packet from the network device 512 in step 6004. In step 6006, the firewall 526 performs standard firewall functions. The firewall 526 then copies the packets and transmits the packet to the destination and the database system 522 in step 6008 before returning step 6002.

FIG. 61 depicts a service capability monitor software architecture for a service capability monitor in an example of the invention. A service capability monitor software architecture 6100 comprises a service broker tokenizer process B 6112, a service broker state map process C 6114, a network A controller process A 6116, a service A signal tokenizer process E 6122, a service A state map process F 6124, a service A controller process D 6126, a service B signal tokenizer process H 6132, a service B state map process F 6134, and a service B controller process D 6136. The network A controller process A 6116 includes a service broker session state objects 6118. The service A controller process D 6126 includes a service A session state objects 6128. The service B controller process D 6136 includes a service B session state object 6138.

The network A controller process A 6116 is connected to the service broker tokenizer process B 6112, the service broker state map process C 6114, the service A controller process D 6126, and the service B controller process D 6136. The service A controller process D 6126 is connected to the service A signal tokenizer process E 6122 and the service A state map process F 6124. The service B controller process D 6136 is connected to the service B signal tokenizer process H 6132 and the service B state map process F 6134.

FIG. 62 depicts a flowchart for the network logic for service capability monitor in an example of the invention. FIG. 62 begins in step 6200. In step 6202, the database system 522 waits for the packet. The database system 522 then receives and processes the packet from the firewall 526 in step 6204. The database system 522 then identifies and authorizes the requester in step 6206. The database system 522 checks if the requester is known in step 6208.

If the requester is known, the network A controller process 6116 writes the packet to process B—service broker signal tokenizer process 6112 in step 6210. The process B 6112 then replies with the function token and parameters to the process A 6116 in step 6212. The database system 522 then checks if a service session state object (SSO) 6118 exists for this session in step 6214. If the SSO 6118 does not exist, the database system 522 creates a SSO 6118 for this session and

53

initializes the current state in step 6216. If the SSO 6118 exists, the database system 522 then proceeds to step 6218. In step 6218, process A 6116 writes the state token and current state to process C—service broker state map process 6114 in step 6218. The process C 6114 then replies with a transition token and parameters to the process A 6116 in step 6220. The process A 6116 applies the transition token and the parameters to the SSO 6118 to update the current state and setting actions in step 6222.

In step 6224, the database system 522 checks if there is an action indicated by the SSO 6118. If there is not an action indicated by the SSO 6118, the database system 522 proceeds to step 6228. If there is an action indicated by the SSO 6118, the database system 522 performs the transition action specified by the SSO 6118 in step 6226. In step 6228, the database system 522 updates the session sliding usage accumulators. The database system 522 then checks if there is an action in the SSO 6118 for this usage in step 6230. If there is no action, the database system 522 returns to step 6202. If there is an action in the SSO 6118, the database system 522 performs the usage action specified in the SSO 6118 before returning to step 6202.

If the requester is not known, the database system 522 checks if there is a network A state object in step 6234. If there is a network state object, the database system 522 proceeds to step 6238. If there is no network state object, the database system 522 creates a network A state object and initializes the current state in step 6236. In step 6238, the database system 522 increments network A state object usage for this event. The database system 522 then checks if there is an action in the network A state object for this usage in step 6240. If there is no action in the network A state object for this usage, the database system 522 returns to step 6202. If there is an action in the network A state object for this usage, the database system 522 performs the usage action specified in the network A state object in step 6242 before returning to step 6242.

FIG. 63 depicts a flowchart for the service logic for service capability monitor in an example of the invention. FIG. 63 begins in step 6300. In step 6302, process D service A controller process 6126 waits for the packet. The process D 6126 then receives and processes the packet from the process A network A controller 6116 in step 6304. The process D 6126 then identifies and authorizes the requester in step 6306. The process D 6126 checks if the requester is known in step 6308.

If the requester is known, the process D 6126 writes the request to process E service A signal tokenizer process 6122 in step 6310. The process E 6122 then replies with the state token and parameters to the process D 6126 in step 6312. The database system 522 then checks if a service session state object (SSO) 6128 exists for this session in step 6314. If the SSO 6128 does not exist, the database system 522 creates a SSO 6128 for this session and initializes the current state in step 6316. If the SSO 6118 exists, the database system 522 then proceeds to step 6318. In step 6218, the process D 6126 writes the state token and current state to process F—service state map process 6124 in step 6318. The process F 6124 then replies with a transition token and parameters to the process D 6126 in step 6320. The process D 6126 applies the transition token and the parameters to the SSO 6128 to update the current state and setting actions in step 6322.

In step 6324, the database system 522 checks if there is an action indicated by the SSO 6128. If there is not an action indicated by the SSO 6128, the database system 522 proceeds to step 6328. If there is an action indicated by the SSO

54

6128, the database system 522 performs the transition action specified by the SSO 6128 in step 6326. In step 6328, the database system 522 updates the session sliding usage accumulators. The database system 522 then checks if there is an action in the SSO 6128 for this usage in step 6330. If there is no action, the database system 522 returns to step 6302. If there is an action in the SSO 6128, the database system 522 performs the usage action specified in the SSO 6128 before returning to step 6302.

If the requester is not known, the database system 522 checks if there is a service A state object in step 6334. If there is a service state object, the database system 522 proceeds to step 6338. If there is no service state object, the database system 522 creates a service A state object and initializes the current state in step 6336. In step 6338, the database system 522 increments service A state object usage for this event. The database system 522 then checks if there is an action in the service A state object for this usage in step 6340. If there is no action in the service A state object for this usage, the database system 522 returns to step 6302. If there is an action in the service A state object for this usage, the database system 522 performs the usage action specified in the service A state object in step 6342 before returning to step 6342.

EXAMPLE

The following is one example of the access communication system that integrates many of the features described in the various embodiments of the inventions above. In the example below, a subscriber called Bank has many employees. The Bank has access to the network architecture 500 through an access provider called AccProv1. In this example, a bank employee is retrieving a stock quote from the yahoo.com website and then switches to request a streaming video from the yahoo.com website. Yahoo's web-server is running in the user network 562. Yahoo has access to the network architecture 500 through an access provider called AccProv2.

A system administrator for AccProv1 configures and optionally starts an access execution environment for the subscriber Bank in the local database system 570. The system administrator also starts a subscriber proxy for the Bank that runs in the access execution environment for the Bank. The subscriber proxy includes transport handlers, configuration handlers, and security handlers. The AccProv1 system administrator starts access execution environments for the maximum number of concurrent users in the local database system 570. The system administrator also starts generic proxies for users in the local database system 570. These generic proxies become user proxies after the user access profile is retrieved for the user.

The system administrator for AccProv2 starts an access execution environment for a service for Yahoo in the local database system 590. The system administrator also starts a service proxy for Yahoo's service that runs in the access execution environment for Yahoo. A system administrator for the Bank then sets up the user access profiles for each employee by inheriting attributes for the user access profile from capability classes or groups that the user belongs to. The user access profile includes the network shell for the user.

The Bank employee then logs on to the access server 524 using a user ID. The Bank system administrator set up. Besides user ID, a user may login to the access server 524 using a prepaid account code, a bank card number, an access card account code, or a user ID including a delimiter for user

access mobility. Upon connection, the access server 524 creates a device session in an access execution environment for the user. Then in response to the user login, the database system 522 retrieves the user access profile for the Bank employee. For a prepaid account code, a bank card number, an access card account code, and a user ID with a delimiter, the database system 522 may retrieve the user access profile from a database system external to the local database system 570. Also, the database system 522 may retrieve the user access profile from a database system external to the local database system 570 for global authentication.

Once the user access profile is retrieved, the access server 524 creates a user session in the access execution environment for the user and binds the device session with the user session. The configuration handler of the user proxy includes the network shell retrieved with the user access profile. The access server 524 then generates and transmits a list of available services in a service based directory to the Bank employee based on the user access profile.

The Bank employee then selects a service to check a stock quote for IBM by transmitting a request for service to the access server 524. The request for service may be in the form of a selection from the service based directory or an alias for alias translation for the network shell using domain name scoping or inbound DNS lookup. The access server 524 processes and transfers the request for service to the user proxy for the Bank employee. The user proxy transfers the request to the service proxy for Yahoo. The service proxy processes and transfers the request for service to Yahoo's web server. The database system 522 translates the service request for Yahoo to a private fulfillment address for one of Yahoo's web servers. The access server 554 then determines if a private destination address is available. If the private destination address is unavailable, the access server 554 could perform many actions including forwarding the request to another Yahoo web server or replying with a busy flag. The access server 524 also binds the user session and the device session with the selected service session by exchanging reference ID's.

The Bank employee then requests another stock quote for Oracle. The access server 524 then determines this stock quote was already cached based on the user's predictable pattern of requesting quotes for this company or a group of software companies. These prior requests for stock quotes of software companies by the Bank employee were audited to derive the Bank employee's predictable pattern. Alternatively, the request for the stock quote for Oracle could have been setup by the user in a script of commands to cache the quote in advance.

The Bank employee then requests a streaming video from Yahoo. The access server 524 switches the access for a premiere access based on the request for streaming video from the Bank employee. Alternatively, Yahoo initiates a switch of access for toll-free Internet service to the Bank employee who is a preferential customer. In order to be able to run the streaming video, the database system 522 also inherits user profile information from a class for streaming video user and updates the user access profile with the user profile information. The user proxy transfers the request for streaming video to the subscriber proxy. The subscriber proxy transfers the request to the service proxy. The service proxy then transfers the request to the Yahoo for one of their streaming video servers. Yahoo's streaming video server then provides the streaming video to the Bank employee.

Conclusion

The access network operates as described in response to instructions that are stored on storage media. The instruc-

tions are retrieved and executed by a processor in the access network. Typically, the processor resides in a server or database. Some examples of instructions are software, program code, and firmware. Some examples of storage media are memory devices, tape, disks, integrated circuits, and servers. The instructions are operational when executed by the processor to direct the network to operate in accord with the invention. The term "processor" refers to a single processing device or a group of inter-operational processing devices. Some examples of processors are computers, integrated circuits, and logic circuitry. Those skilled in the art are familiar with instructions, processors, and storage media.

Those skilled in the art will appreciate variations of the above-described embodiments that fall within the scope of the invention. As a result, the invention is not limited to the specific examples and illustrations discussed above, but only by the following claims and their equivalents.

1 claim:

1. A method of operating an access system including an access server to provide access between a user system and a plurality of communication networks that provide services to a user, the method comprising:

receiving a user login into the access server;

processing the user login to determine if the user is allowed access to the access system based on a local database system;

providing access to the access system to the user in response to the determination that the user is allowed access based on the local database system;

generating an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system;

in the second database system, receiving and processing the authorization query to determine whether the user is allowed access;

in the second database system, generating and transmitting an authorization response to the local database system;

receiving and processing the authorization response indicating whether the user is allowed to use the access system from the second database system; and

providing access to the access system to the user in response to the authorization response that allows the user to use the access system.

2. The method of claim 1 further comprising generating and transmitting a login query.

3. The method of claim 1 wherein processing the user login to determine if the user is allowed access to the access system based on a local database system is based on a user access profile.

4. The method of claim 1 further comprising determining if the second database system exists.

5. The method of claim 1 further comprising disconnecting a user's network device from the access server in response to the determination that the user is not allowed to use the access system.

6. The method of claim 1 wherein the login reply includes an access card account code.

7. The method of claim 1 wherein the login reply includes foreign network account information and further comprising identifying the second external database based on the foreign network account information.

8. The method of claim 1 further comprising:

receiving a request for access into a service control point;

57

processing the request for access to determine the destination for the request;

generating and transmitting a reply to route the request to the access server;

9. The method of claim 1 further comprising logging contract and settlement information.

10. An access system for providing access between a user system and a plurality of communication networks that provide services to a user, the access system comprising:

an access server connected to the user system and the plurality of communication networks and configured to receive and transmit a user logon from the user system to a local database system;

the local database system connected to the access server and configured to receive the user logon, process the user logon to determine if the user is allowed access to the access system based on the local database system, provide access to the access system to the user in response to the determination that the user is allowed access based on the local database system, generate an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system, receive and process an authorization response indicating whether the user is allowed to use the access system from the second database system, and provide access to the access system to the user in response to the authorization response that allows the user to use the access system; and

the second database system connected to the local database system and configured to receive and process the authorization query to determine whether the user is allowed access and generate and transmit the authorization response for the local database system.

11. The access system of claim 10 wherein the access server is configured to generate and transmit a logon query.

12. The access system of claim 10 wherein the local database system is configured to process the user logon to determine if the user is allowed access to the access system based on a user access profile.

13. The access system of claim 10 wherein the local database system is configured to determine if the second database system exists.

14. The access system of claim 10 wherein the access server is configured to disconnect a user's network device in response to the determination that the user is not allowed to use the access system.

15. The access system of claim 10 wherein the logon reply includes an access card account code.

16. The access system of claim 10 wherein the logon reply includes foreign network account information and the local database system is configured to identify the second external database based on the foreign network account information.

58

17. The access system of claim 10 further comprising:

a service control point connected to the user system configured to receive a request for access, process the request for access to determine the destination for the request, and generate and transmit a reply to route the request to the access server.

18. The access system of claim 10 wherein the local database system is configured to log contract and settlement information.

19. A software product for providing access between a user system and a plurality of communication networks that provide services to a user, the software product comprising:

database software operational when executed by a processor to direct the processor to receive the user logon, process the user logon to determine if the user is allowed access to an access system based on a local database system, provide access to the access system to the user in response to the determination that the user is allowed access based on the local database system, generate an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system, receive and process an authorization response indicating whether the user is allowed to use the access system from the second database system, and provide access to the access system to the user in response to the authorization response that allows the user to use the access system; and

a software storage medium operational to store the database software.

20. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to process the user logon to determine if the user is allowed access to the access system based on a user access profile.

21. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to determine if the second database system exists.

22. The software product of claim 19 wherein the logon reply includes an access card account code.

23. The software product of claim 19 wherein the logon reply includes foreign network account information and the database software is operational when executed by the processor to direct the processor to identify the second external database based on the foreign network account information.

24. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to log contract and settlement information.

* * * * *

U.S. PATENT DOCUMENTS

6,195,697 B1	2/2001	Bowman-Amuah	6,272,152 B1	8/2001	Levin et al.
6,208,720 B1	3/2001	Curtis et al.	6,282,276 B1	8/2001	Felger
6,212,262 B1	4/2001	Kamel	6,289,010 B1	9/2001	Voit et al.
6,233,313 B1	5/2001	Farris et al.	6,295,292 B1	9/2001	Voit et al.
6,247,047 B1	6/2001	Wolff	6,330,543 B1	12/2001	Kepecs
6,260,024 B1	7/2001	Shkedy	2002/0147658 A1	10/2002	Kwan

* cited by examiner

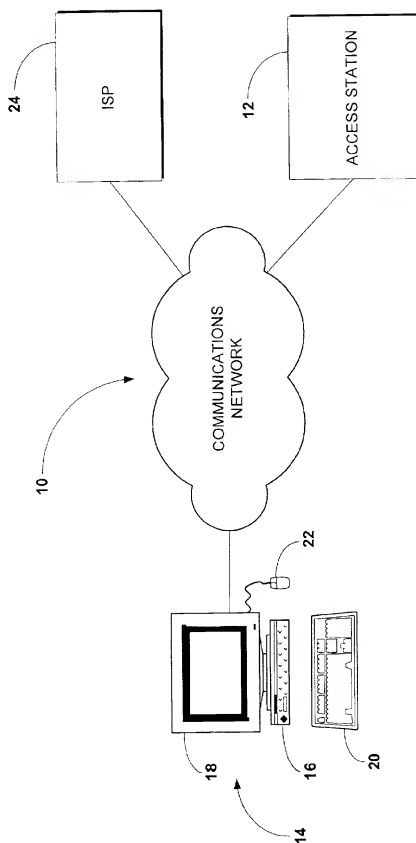


Figure 1.

Figure 2a.

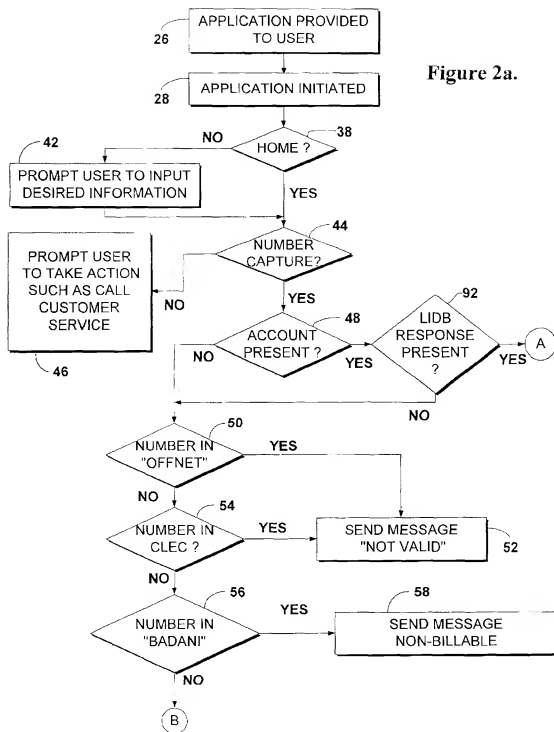
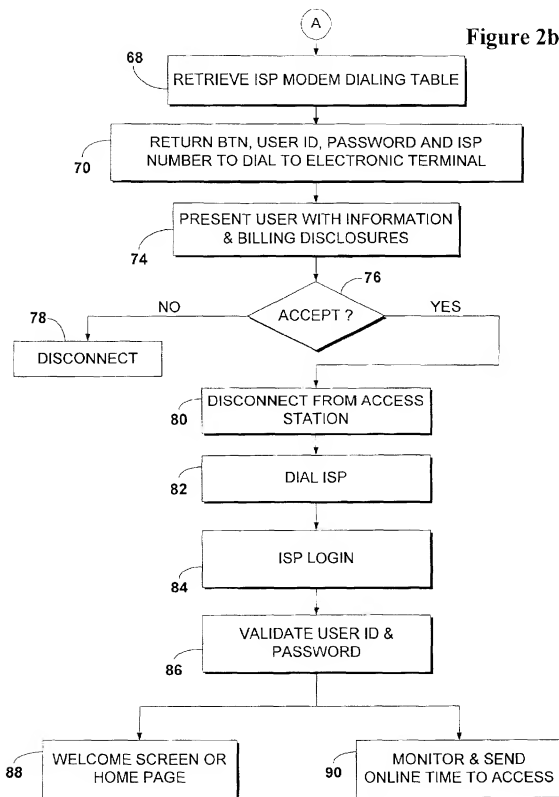
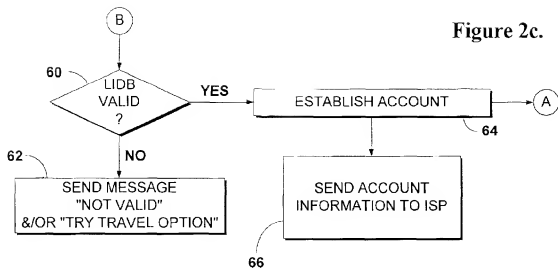


Figure 2b.





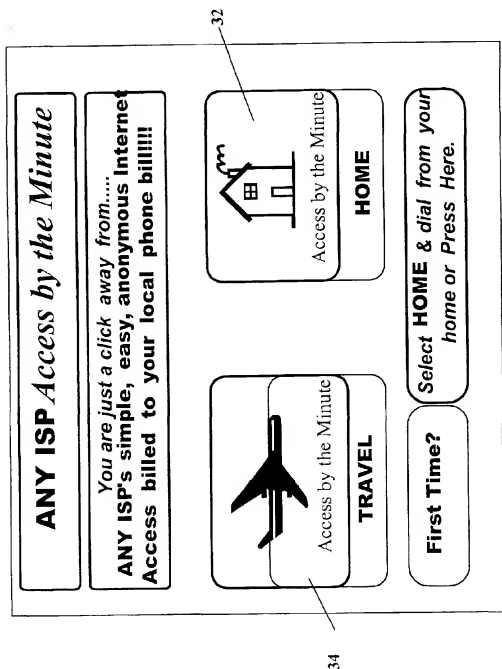


Figure 3a.

ANY ISP *Access by the Minute*

First Time???

Please ensure you:

- are dialing from your home phone
- are authorized to charge to this phone number
- understand this call will be charged per minute

CONTINUE

BACK

Figure 3b.

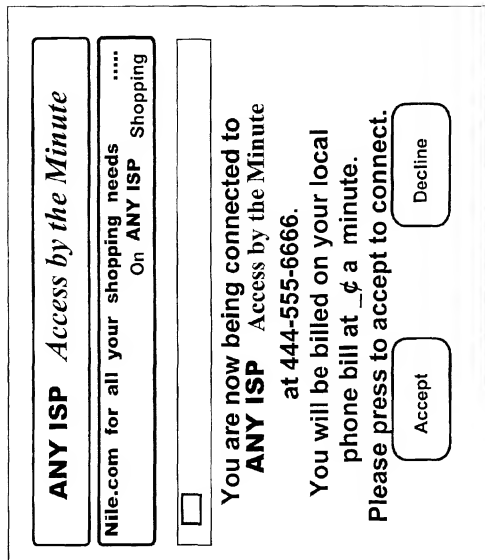


Figure 3c.

PACIFIC BELL

Account Number 408 229 765 N 6159 Statement Date Dec 3, 1998 Page 11
 Questions about your bill? 1-800-736-7900



Total Current Charges (See detail below)

Monthly Charges

Description	Rate	Local	Amount
1. Direct Dial	.60	911	1.60
Total Monthly Charges			\$1.60

Calls from 408 XXX-XXXX Billed on Behalf of AOL Access By The Minute Direct Dialed ISP Access Calls:

Date & Time	Place and Number Called	Type	Rate	Minutes	Amount
6 Feb 10 2:17pm	444 555-7777	Direct	Day	6	.42
7 Feb 11 12:30pm	444 555-7777	Direct	Day	10	.70
8 Feb 12 12:30pm	444 555-7777	Direct	Night	7	.49
9 Feb 13 12:30pm	444 555-7777	Direct	Day	8	.56
10 Feb 16 2:56pm	444 555-7777	Direct	Day	8	.56
11 Feb 17 3:46pm	444 555-7777	Direct	Day	13	.84
12 Feb 22 3:46pm	444 555-7777	Direct	Day	5	.36
13 Feb 23 12:30pm	444 555-7777	Direct	Day	5	.36
14 Feb 23 8:34pm	444 555-7777	Direct	Day	5	.36
15 Feb 23 8:34pm	444 555-7777	Direct	Day	46	3.22
Total ISP Access Calls					\$16.90

Taxes & Surcharges

Description	Amount
16 Charges for Network Access for Interstate Calling	3.00
17 CA High Cost Fund Surcharge A	.07
18 CA High Cost Fund Surcharge B	.07
19 Rate Surcharge	1.34
20 State Regulatory Fee	.03
21 State Regulatory Fee	.03
22 Tax, Fed, State, 99 911	1.50
Total Taxes & Surcharges	\$4.31

13 6291 B251 1A 408 2244657 765 9613818E7 R001 RTN 9273

Figure 4.

1

SYSTEM AND METHOD FOR ACCESSING THE INTERNET ON A PER-TIME-UNIT BASIS

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention is generally directed to a system and method for accessing a global electronic communications network. More particularly, the present invention is directed to a system and method for accessing the Internet from an electronic terminal, wherein an account associated with the electronic terminal is billed in monetary units corresponding to the length of time the electronic terminal is connected to the Internet.

2. Description of the Related Art

Internet access has conventionally required users to obtain a subscription from an Internet service provider. The form of such subscriptions vary, depending upon the provider and the quantity of internet access desired. For example, one conventional approach is to charge a subscriber a fixed monetary amount for unlimited use of the Internet, via the provider, during a predetermined period, such as a month. In variations of this approach, a user may be charged a first fixed amount for a first amount of access time during a given period, and then additional amounts for access over the initially allotted time. Charges for Internet access are typically billed to a financial account, such as a credit card of the subscriber.

One drawback to conventional approaches for providing an Internet access is the requirement is that the user establish a subscription and, particularly, the requirement that a user enter financial account information in order to establish the subscription. While electronic transactions over the Internet are becoming increasingly common place, a significant number of individuals, and especially those that may remain skeptical of security issues or, alternatively, those that simply have had little or no exposure to the Internet, may remain unwilling to submit financial information to an Internet service provider in order to establish the subscription. Additionally, those users that access the Internet infrequently or sporadically may be unwilling to incur periodic subscription charges.

Accordingly, the need exists for a system and method for accessing the Internet that provides an alternative to existing Internet access arrangements. In particular, the need exists for a system and method for accessing the Internet in a manner which does not require a conventional subscription. The present invention fulfills these and other needs.

SUMMARY OF THE INVENTION

The present invention is a system and method for accessing global electronic communications network and, particularly, for accessing the Internet. The system of the present invention has an Internet access station, and a site associated with an Internet service provider. Using an electronic terminal, such as a personal computer, or a fixed, portable, or mobile computing station, a user may access the Internet via the Internet service provider, albeit in accordance with the unique aspects of the present invention.

In accordance with one aspect of the present invention, the access station is a site at which various processes are performed as a precondition to permitting the electronic terminal to establish a communications link with the Internet service provider. Additionally, the access station also pref-

2

erably serves as a billing station, such that charges associated with Internet access with the electronic terminal, through the Internet service provider, are billed to the user of the electronic terminal via the access station.

In particular, in accordance with the principles of the present invention, application software for permitting the user of the electronic terminal to access the Internet are provided to the electronic terminal. The application software permits a user of the electronic terminal to establish a communications link with the access station. Once a communications link with the access station is established, the access station, utilizing a processor, processes a series of steps in order to automatically establish an account associated with the electronic terminal. Once the account is established, information indicative of the account is transmitted via the communications link back to the electronic terminal along with connection information, such as a telephone number, for enabling the electronic terminal to initiate a second communications link with the Internet service provider. Additionally, information indicative of the established account is transmitted via a third communications link from the access station directly or at scheduled intervals to the Internet service provider, where it is stored as a profile.

Upon receipt by the electronic terminal of the account information and connection information, the first communications link between the electronic terminal and the access station is terminated, and the electronic terminal (using the connection information received from the access station) initiates a second communications link with the Internet service provider. Once the second communications link between the electronic terminal and Internet service provider is established, the electronic terminal transmits the account information received from the access station to the Internet service provider. The Internet service provider processes that information by comparing it with the short profile information received from the access station, in order to confirm that the electronic terminal seeking access is a valid terminal for which access should be permitted. Upon completion of that validation process, the electronic terminal is logged on to the Internet via the Internet service provider.

In accordance with an aspect of the present invention, upon initiation of the second communications link between the electronic terminal and the Internet service provider or, at some other convenient time, such as upon a log on time to the Internet, the Internet service provider begins monitoring the time of the second communications link (or log on time). That time duration is monitored in time units, such as minutes, and is stored in an electronic memory of the Internet service provider for periodic transmission directly to the access station. It will be understood and appreciated that while the time of the communications link or Internet access time is preferably transmitted from the Internet service provider to the access station periodically, it may be continuously transmitted via a direct communications link between the Internet service provider and access station. At the access station, the time units associated with the second communications link, or log in time, is associated with the account established for the electronic terminal, and a monetary rate is multiplied by the time units for billing purposes. In particular, although the monetary charges may be calculated at the Internet service provider, these calculations may be made at the access station by multiplying the number of monetary units associated with a particular log in session by a published rate, and associating the resulting charges with the established account.

In accordance with a particular aspect of the present invention, the access station automatically determines the

3

telephone number from which the electronic terminal is accessing the access station. In a preferred embodiment, the determined telephone number is utilized as a billing number in the process of establishing the account. Thus, in accordance with the invention, any charges associated with Internet access per time unit are billed to an account or subscription associated with the telephone number from which the electronic terminal accesses the access station. Particularly, the charges are consolidated on a telephone bill which includes other charges corresponding to the telephone number, such as local and long distance telephone calls.

Additionally, before permitting the electronic terminal to link with the Internet service provider, the access station preferably performs various validation processes for determining whether the telephone number associated with the electronic terminal is acceptable for billing purposes. For example, the access station preferably determines whether the telephone number is within one or more selected networks, and determines whether there are other acceptable credit risks associated with the telephone number. Additionally, the access station preferably retrieves information from a line information data base (LIDB) to ascertain whether the telephone number is associated with a valid, existing telephone line subscription. These validation features are provided upon initial access by the electronic terminal to the access station, but are preferably only provided on subsequent Internet access attempts when the time lapse between access attempts is greater than a predetermined period, such as for example, 30 days.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects and features of the invention noted above are explained in more detail with reference to the drawings, in which like reference numerals denote like elements, and in which:

FIG. 1 is a block diagram of the Internet access system of the present invention;

FIGS. 2a-2c illustrate a flow chart illustrating the method of the present invention and the software processes performed by the present invention;

FIGS. 3a-3c are examples of screen displays provided on the display screen of the electronic terminal utilized for accessing the Internet in accordance with the principles of the present invention; and

FIG. 4 is an example of monetary charges incurred for accessing the Internet in accordance with the principles of the present invention, wherein those charges are consolidated on a standard telephone bill.

DETAILED DESCRIPTION OF THE INVENTION

With reference initially to FIG. 1, an Internet access system of the present invention is denoted generally by reference numeral 10. The Internet access system 10 of the present invention has an Internet access station, denoted by reference numeral 12, which serves a variety of functions, as described in detail below. In particular, Internet access station 12 is preferably a site on a global communications network at which an Internet access account is established and maintained, at which Internet access validation procedures are conducted and at which various billing activities are processed. The Internet access system 10 further includes an electronic terminal 14. Electronic terminal 14 is illustrated in FIG. 1 as a personal computer having a processor 16, a display 18, a keypad input 20, and a mouse 22. It will

4

be understood and appreciated that the electronic terminal 14 may include other desired components, such as microphone input, speakers, a scanner, a web camera, etc. Additionally, it should be understood and appreciated that the electronic terminal 14, rather than being a conventional personal computer, may be any type of electronic terminal for accessing the Internet. For example, the electronic terminal 14 may be a portable or laptop computer, or a mobile or wireless Internet equipped communications device, such as a cellular telephone.

Internet access system 10 further includes an Internet service provider, denoted generally by reference numeral 24. Internet service providers are well known. In particular, Internet service provider 24 is a site, at an address, on a global communications network. In particular, Internet service provider 24 is a site which serves as a portal through which users of electronic terminals may access the Internet. As illustrated, a communications network is provided for establishing communications links between electronic terminal 14, access station 12, and Internet service provider 24. The communications network may be any conventional type of network, such as a telephony network, or a data network, may be a broadband (or not) network, etc. or any combination thereof. Access to the network, through the various components of the present invention, may be made in a conventional manner through interface components, such as a modem or transceiver.

With additional reference to FIGS. 2a-2c, the system 10 and its method of operation will be described in detail.

In accordance with an aspect of the present invention, and as indicated at step 26 in FIG. 2a, an Internet access software application is provided to electronic terminal 14 in any conventional manner. For example, a user of electronic terminal 14 may download the necessary application software from access station 12 or Internet service provider 24. Alternatively, the application software may be provided on a transferable medium, such as a CD, with which the electronic terminal 14 communicates. Thus, as will be understood and appreciated, the application software may be stored on either a portable, transferable medium, or may be stored on a hard drive memory within the electronic terminal 14. Alternatively, the Internet access application software of the present invention may be provided in an open system arrangement, in which the Internet access software of the invention is provided at the access station 12, such that electronic terminal 14 communicates with the application software via the communications network, although the application software is not physically stored at the electronic terminal 14.

As indicated at reference numeral 28, a user of electronic terminal 14 initiates the Internet application software of the present invention in any conventional manner, such as by utilizing computer mouse 22 to click on an icon associated with the software displayed on display screen 18. Upon initiation of the application software at step 28, electronic terminal 14 displays at display 18 a display screen, such as that illustrated in FIG. 3a. In particular, the display screen provides the user of electronic terminal 14 with several options. For example, the display screen may provide an area for accessing the Internet from a home location as indicated by area 32 on the display screen, or for accessing the Internet from a remote, travel location, as indicated by area 34 on the display screen. The present invention may also provide to the user the option to access a "first time ?" interface, whereupon the user will be taken to a page providing various comments and instructions relating to first time use. For example, as illustrated in FIG. 3b, if the user

5

clicks on an area associated with "first time" use, the application may provide an individual with detailed instructions regarding the various choices available to the user. Additional information, such as indications that the user will be charged to his or her local phone bill on a per-time basis for Internet access, along with prompts requiring user entry to indicate whether or not the user wishes to continue may be included.

Additionally, the application prompts the user to input various information, such as the telephone number from which dialing is taking place, whether call waiting is a feature (so that it may be disabled, if necessary), and other dialing information (e.g., 9 or 8 access, area code information, etc.). Alternatively, some or all of the input information may be captured automatically, such as with a caller-id feature, etc. Additionally, all gathered data is stored in memory by the application for retrieval upon subsequent uses.

In short, regardless of where the information is provided (e.g., on the opening page, or after accessing one or more pages associated with a "help" feature, or a "first time" feature) when the user initially utilizes system 10, and particularly the software application thereof, the user is prompted to proceed via the "home location" area 32. As indicated at step 38 of FIG. 2, when the user selects Internet access from the home location, processing advances to step 42 for initiating a communications link with access station 12. Alternatively, as will be described in greater detail below, when the user desires to access the Internet from a non-home location, such as is the case when a user has a portable or laptop computer, and, for instance, is traveling and is thus attempting to access the Internet through a telephone line other than the user's home telephone line, the processing advances to step 40, wherein the user is prompted to enter additionally required information. In such an instance, for example, the user may be prompted to input additional information, such as the type of communications line relied upon at the remote location (e.g., pay phone, air phone, hotel, work, someone else's home, etc.). Additionally, information pertaining to the telephone number from which access is being desired may also be required, such as specific dialing instructions necessary from dialing at a remote location (e.g., the requirement to dial a "8" prior to dialing from a hotel, etc.). In either event, electronic terminal 14 may initiate a communications link by dialing a telephone number of the access station (e.g., a toll free or non-toll free number, including but not limited to a 1-800 or 1-900 number), or by otherwise linking with a server associated with an electronic address of the access station.

Once the electronic terminal 14 establishes a communications link with the access station 12, the access station 12, utilizing a processor and software, executes a number of processing steps and makes a number of determinations. In particular, upon the initial time the electronic terminal 14 links with the access station, the access station performs a number of validation functions in an effort to establish an account associated with the electronic terminal 14. As will become apparent from the detailed discussion below, upon subsequent links with access station 12, various validation features are performed, but the account need not be reestablished.

In particular, once a communications link is established between the electronic terminal 14 and the access station 12, an automatic number identification (ANI) process occurs, in which the access station 12 determines the telephone number from which the electronic terminal 14 is communicating. As indicated at step 44 of FIG. 2, access station 12 deter-

6

mines whether the telephone number from which the electronic terminal 14 is dialing has been captured. As indicated at step 46, in the event the telephone number has not been captured, the access station 12 transmits a message back to the electronic terminal 14 prompting the user to action such as call customer service. Alternatively, once the telephone number from which electronic terminal 14 is dialing has been captured and stored at the access station 12, the access station 12 determines whether an account has already been established with respect to the captured telephone number. As will be described below, in the process of initially establishing an account, the access station initially establishes the captured telephone number as a billing telephone number to which charges associated with Internet access time will be billed. As additionally described below, the access station also assigns a user identification and password to the billing telephone number, thereby establishing the account. Accordingly, at step 48, where it is determined whether an account has already been established with respect to the captured telephone number, the processor at access station 12 searches an associated database which stores account information to determine whether the captured telephone number has been assigned to an associated billing telephone number and/or whether the telephone number has an associated user identification and/or password.

In the event an account has not already been established with respect to the captured telephone number, such as is the case when electronic terminal 14 is initially utilized to access the Internet via the system 10 of the present invention, processing advances to step 50. At step 50, the access station 12 determines whether the captured telephone number (e.g., "ANI") is in a particular network that is not affiliated with the access station. In other words, there may be various communications networks with which billing arrangements have not been established, and a determination is made at step 50 whether the captured telephone number is within a predetermined network. In particular, as illustrated, the determination is made whether the captured telephone number is within "OFFNET", which is intended to identify an off network (e.g., one with which a billing arrangement has not been established). It will be understood and appreciated that the determination made may be whether the captured telephone number is within a non-acceptable network or, alternatively, by making a determination whether the captured telephone number is within an acceptable network. As illustrated, in the event the captured telephone number is in a non-acceptable network, such as the case when the captured telephone number is in "OFFNET", processing advances to step 52, wherein the access station 12 sends a message back to the electronic terminal 14 indicating that the telephone number from which access is desired is not valid. Alternatively, when it is determined that the captured telephone number is not within "OFFNET", and therefore may be acceptable, processing advances to step 54, where access station 12 determines the captured telephone number is within a CLEC (competitive local exchange carrier) network database, thus preventing the billing of Internet access charges directly to a telephone bill.

As illustrated, if the captured telephone number is within the CLEC database, processing advances to step 52, and under such circumstances, a message is transmitted from access station 12 to electronic terminal 14 indicating that the telephone number from which Internet access is desired is not valid. However, if the captured telephone number is not within the CLEC database, as determined at step 54, processing advances to step 56, where access station 12 deter-

7

mines whether the captured telephone number has an unacceptable associated credit risk. In this regard, a data base of telephone numbers associated with certain credit risk standards is maintained, such that the captured telephone number is checked against those telephone numbers accumulated in the data base, in an effort to determine whether the captured telephone number is associated with an unacceptable or questionable credit risk. As illustrated, if it is indeed determined that the captured telephone number is associated with an unacceptable or questionable credit risk, processing advances to step 58, wherein the access station transmits a message to the electronic terminal 14 indicating that the telephone number from which access is desired is potentially non-billable, and may further request user action such as a user to telephone the access station 12. However, in the event customer service 12 determines at step 56 that the captured telephone number is not associated with an undesirable or questionable credit risk (e.g., not in a "BADANI" database as illustrated), processing advances to step 60.

At step 60, access station 12 determines whether the captured telephone number is a valid telephone number. In other words, access station 12 checks the telephone number against a line information database (LIDB) in making a validity decision with respect to the captured telephone number. As indicated, in the event a LIDB check results in a determination that the captured telephone number is not valid, processing advances to step 62, and access station 12 sends the electronic terminal 14 a message that the telephone number from which access is desired is not valid, and prompts the user to consider take action such as a call to customer service.

When, however, it is determined at step 60, as a result of the LIDB check, that the captured telephone number is indeed a valid telephone number, processing then advances to step 64, where access station 12 considers the captured telephone number to be a billable telephone number (BTN) (e.g., the number to which Internet access charges will be billed), and access station 12 automatically assigns a user identification and password to the billing telephone number to thereby establish an account with respect to the captured telephone number from which electronic terminal 14 is attempting to access the Internet. Access station 12 then stores the BTN, and associated user ID and password in a database. As indicated at step 66, access station 12 also then sends at that time or at scheduled intervals, via a direct communications link with Internet service provider 24, the account information that has been established with respect to electronic terminal 14. In particular, access station 12 sends to ISP 24 at least the user identification and password associated with the established account. As will become apparent from the following discussion, the Internet service provider 24 will subsequently use the retrieved account information to validate an attempt by the electronic terminal 14 to access the Internet via the ISP 24.

Additionally, once an account has been established at step 64, the processing advances to step 68, where the access station 12 retrieves a table of identifiers (e.g., telephone numbers or server addresses, associated with Internet service provider 24). In other words, the retrieved table includes, for example, modem dialing information associated with Internet service provider 24.

Processing then advances to step 70 at which the access station 12 transmits the established account information, such as the billable telephone number (BTN), user identification, password, and at least one telephone number or address associated with the ISP 24, and retrieved from the dialing table, for the purpose of permitting the electronic

8

terminal 14 to establish a communications link with ISP 24. In particular, and in accordance with the principles of the present invention, the identifier or identifiers (e.g., the telephone number or numbers retrieved from the ISP dialing table), are preferably determined based upon a location of the telephone number from which access is being allowed or from the location of the billable telephone number and, additionally, in the case where multiple identifiers are retrieved, they are preferably prioritized in accordance with selected criteria. In particular, local telephone numbers are given priority.

At step 70 once the access station 12 has returned the billable telephone number, user ID, password, and one or more identifiers (e.g., telephone numbers or addresses) of the ISP 24 to the electronic terminal 14, processing advances to step 74. At step 74, the user is presented with information and billing disclosures (which may be retrieved from the electronic terminal 14, but which are preferably transmitted from access station 12). As indicated at 76, the user is prompted to accept or not accept the proposed billing agreement. An example of a display screen associated with step 76 is illustrated in FIG. 3c. In this regard, and in accordance with an aspect of the invention, the information provided by the billing disclosure indicates that the user will be charged to his or her home telephone number (e.g., the telephone number from which electronic terminal 14 accesses station 12), and the user is given rate information, (such as monetary charges per minute). For example, the stated charges may be a number of cents (USA) per minute. Alternatively, the charges could be per second, per six second increments, or in other increments. As indicated at step 78, if the user elects to deny the billing proposal, processing advances to step 78 and the communications link with the access station 12 is terminated. Alternatively, when the user, at step 76, accepts the billing proposal, processing advances to step 80 and then to step 82, where the communications link between the electronic terminal 14 and access station 12 is terminated, and the electronic terminal 14 initiates a communications link with ISP 24 by dialing ISP 24, respectively. As indicated at step 86, once a communications link has been established between electronic terminal 14 and the ISP 24, the processor at ISP 24 makes a determination of whether the electronic terminal 14 desiring to access ISP 24 is valid. In particular, the electronic terminal 14 sends, via the communications link with ISP 24, the account information that the electronic terminal 14 previously received from access station 12. In particular, electronic terminal 14 transmits data indicative of at least the user identification and password, and the billable telephone number may also be transmitted. A processor at the ISP 24 retrieves the account information data and compares it with the data previously transmitted from access station 12 to determine whether the electronic terminal 14 is associated with a "valid" account. When the ISP 24 is unable to read the account information transmitted by electronic terminal 14, or in the situation where the received account information is not found in the data base of accounts deemed to be valid, log onto the ISP 24 is denied. However, as indicated at step 88, once the account associated with electronic terminal 14 is properly validated, ISP 24 opens a welcome screen or home page, from which the user can browse the Internet or access one or more selected features presented to the user in a conventional fashion.

Additionally, ISP 24 monitors a time associated with a communications link between electronic terminal 14 and ISP 24. In this regard, while the time monitored may encompass the entirety of the time associated with a com-

9

munications link, preferably ISP 24 begins monitoring time only after log on validation procedures have been conducted and the user of electronic terminal 14 is actually logged in to ISP 24. While charges for Internet access time preferably begin after log in, any other communications charges may begin at the establishment of the communications link. It should be understood that the access time may alternatively be monitored from a time associated with access to particular content (e.g., news, video, etc.) or that different or increased rates may apply to the access or downloading of selected content.

Preferably, as indicated at step 90, ISP 24 periodically sends information to access station 12 regarding the amount of time electronic terminal 14 accesses ISP 24. It will be understood and appreciated that this on-line time information, transmitted from ISP 24 to access station 12, is preferably transmitted directly from ISP 24 to access station 12, and not via electronic terminal 14. Additionally, it will be understood and appreciated that the on-line time information may be transmitted from ISP 24 to access station 12 at any preferable interval (such as hourly, daily, etc.). Additionally, the intervals may be varied such that the information is not transmitted at the same time each day.

Returning now to step 48 of FIG. 2, when it is determined that the identifier (e.g., telephone number or server address) is already associated with an established account, processing advances to step 92 at which access station 12 makes a determination whether a previous LIDB check (e.g., that check made at step 60) has been made within a predetermined time frame. For example, as illustrated, the access station 12 queries whether a LIDB response related to the BTN is present. As indicated, if a LIDB response is present in the database, processing advances directly to step 68, and the ISP modem dialing table is retrieved, and processing continues in the manner illustrated and previously described. Alternatively, however, when a determination is made at step 92 that a LIDB response is not present in database, processing advances to step 50, and the illustrated and previously described tree following step 50 is executed. In other words, the step or steps associated with steps 50, 54, 56 and/or 60 are processed, thereby providing a validation procedure.

With reference now to FIG. 4, an example of an invoice sent to the user of electronic terminal 14 is illustrated. In particular, the access station 12, upon receipt of on-line time associated with electronic terminal 14, stores data indicative of the on-line time in the established account corresponding to electronic terminal 14. Accordingly, stored in the account associated with electronic terminal 14 is, for each access event, a date of the access event, a time of day of the access event, the time associated with the access event, and accumulative charge associated with the access event. Additionally, other information associated with the place and telephone number called to achieve the access, rate information, etc., may also be stored in the account in conjunction with each access event. Then, at the end of a selected billing cycle, such as a one month billing cycle, data is retrieved and placed on an invoice along with other charges associated with the billable telephone number, such as long distance charges, taxes, etc. Alternatively, the information retrieved from the Internet access account associated with electronic terminal 14 may be transmitted to another billing entity, for consolidation on a printed or electronic invoice.

From the foregoing, it is seen that the present invention is a highly useful system for accessing the Internet via an electronic terminal, where charges associated with Internet

10

access time are billed directly to a bill associated with the electronic terminal, or the telephone line from which the terminal accesses the Internet. Additionally, the system 10 is useful for approving access only to those electronic terminals 14 or individuals which have predetermined criteria. In other words, access station 12 is useful for preventing access where statistical information indicates that the terminal from which access is desired may be associated with an undesirable credit risk or, alternatively, where the terminal from which access is desired is within a particular network such that billing will be cumbersome or possible, or where the telephone line associated with the electronic terminal is otherwise not a valid line.

Additionally, as will be understood and appreciated, in a preferred embodiment of the present invention, a multitude of electronic terminals 14 are utilized to access the Internet via system 10. In other words, proprietors of access station 12 will have relationships with a wide variety of carriers associated with the communications network, such that a wide variety of individuals may utilize an electronic terminal for accessing the ISP via the access station 12, and thereafter be billed on a per time basis, for the Internet access, preferably on an invoice directly associated with a particular carrier or telephone line.

From the foregoing it will be seen that this invention is one well adapted to attain all ends and objects hereinabove set forth together with the other advantages which are obvious and which are inherent to the structure.

It will be understood that certain features and subcombinations are of utility and may be employed without reference to other features and subcombinations. This is contemplated by and is within the scope of the claims.

Since many possible embodiments may be made of the invention without departing from the scope thereof, it is to be understood that all matter herein set forth or shown in the accompanying drawings interpreted as illustrative, and not in a limiting sense.

What is claimed is:

1. A system for accessing an electronic network via a service provider, said system comprising:

- an electronic terminal having an associated identifier;
- an access station which establishes an account and transmits connection information back to said electronic terminal, wherein the account is established based upon an investigation of a local exchange carrier database;
- a first communication link to link said electronic terminal and said access station wherein said identifier associated with said electronic terminal is transmitted to said access station; and
- a second communications link between said electronic terminal and said service provider, said second communications link initiated between said electronic terminal and said service provider based upon said connection information,

wherein said service provider monitors a time of said second communications link and transmits data indicative of said time to said access station.

2. The system as set forth in claim 1, where in said access station associates said time with said account for billing purposes.

3. The system as set forth in claim 2, wherein said identifier of said electronic terminal is a telephone number.

4. The system as set forth in claim 3, wherein said electronic terminal is portable, and when said electronic terminal is used at a remote location from a home port associated with said telephone number, said access station requires entry of additional information.

11

5. The system as set forth in claim 4, wherein said additional information corresponds to said account.

6. The system as set forth in claim 4, wherein said additional information is indicative of dialing instructions at said remote location.

7. The system as set forth in claim 3, wherein a monetary amount corresponding to said time of said second link is charged to said telephone number.

8. The system as set forth in claim 1, wherein said access station associates a task with said identifier of said electronic terminal, wherein a password is transmitted to said electronic terminal via said first communications link and to said electronic network service provider via a third communications link.

9. A system for accessing an electronic network via an electronic network service provider, said system comprising:

an electronic terminal having an associated identifier and configured to access said electronic network;

an access station, configured to link with said electronic terminal, to store data indicative of said identifier, and to provide said electronic terminal with information associated with said electronic network service provider for use by said electronic terminal in linking with said electronic network service provider responsive to an investigation of a local exchange carrier database, wherein, once a link between said electronic terminal and said electronic network service provider is established, said electronic network service provider monitors a number of time units associated with said established link and transmits data indicative of said time units to said access station for use in billing matters.

10. The system as set forth in claim 9, wherein said identifier associated with said electronic terminal is a telephone number associated with a subscriber.

11. The system as set forth in claim 9, wherein said identifier is an electronic address associated with a subscriber.

12. A method of accessing the Internet with an electronic terminal and via an Internet service provider (ISP), said electronic terminal having an associated identifier, said method comprising:

initially linking said electronic terminal with an access station via a first communications link;

establishing an account at said access station corresponding to said electronic terminal responsive to an investigation of a local exchange carrier database;

transmitting from said access station to said electronic terminal ISP connection information pertaining to said ISP;

using said connection information to link said electronic terminal with said ISP via a second communications link;

monitoring time units associated with said second communications link at the ISP;

12

multiplying said time units by a monetary rate to thereby obtain billing data; and

associating said billing data with said established account for billing purposes.

13. The method as set forth in claim 12, wherein establishing said account further comprises:

determining an identifier associated with said electronic terminal; and

using said identifier to establish a billing identifier.

14. The method as set forth in claim 13, wherein establishing an account further comprises:

establishing a password for said account.

15. The method as set forth in claim 14, further comprising:

transmitting said billing identifier, said password, and an identifier associated with said ISP to said electronic terminal as part of said connection information.

16. The method as set forth in claim 15, further comprising:

additionally transmitting said billing identifier and said password to said Internet service provider via a third communications link.

17. The method as set forth in claim 16, when said electronic terminal links with said Internet service provider via said second communications link, the method including transmitting said billing identifier and said password from said electronic terminal to said Internet service provider and comparing said billing identifier and said password transmitted via said second link with said billing identifier and password transmitted via said third link for log on purposes at said ISP.

18. A method of accessing the Internet with an electronic terminal and via an Internet service provider, said electronic terminal having an associated identifier, said method comprising:

initially linking said electronic terminal with an access station via a first communications link;

establishing an account at said access station corresponding to said electronic terminal responsive to an investigation of a local exchange carrier database;

transmitting from said access station to said electronic terminal ISP connection information pertaining to said ISP;

using said connection information to link said electronic terminal with said Internet service provider via a second communications link; and

associating billing data with said established account at said ISP for billing purposes.

19. The method of claim 18, wherein said identifier is a telephone number.

* * * * *



US006636259B1

(12) **United States Patent**
Anderson et al.

(10) **Patent No.:** **US 6,636,259 B1**
(45) Date of Patent: **Oct. 21, 2003**

(54) **AUTOMATICALLY CONFIGURING A WEB-ENABLED DIGITAL CAMERA TO ACCESS THE INTERNET**

6,167,469 A * 12/2000 Safai et al. 710/62
 6,226,752 B1 * 5/2001 Gupta et al. 713/201
 6,269,481 B1 * 7/2001 Perlman et al. 717/11
 6,502,195 B1 * 12/2002 Colvin 713/202

(75) Inventors: **Eric C. Anderson**, San Jose, CA (US);
Robert Paul Morris, Raleigh, NC (US)

* cited by examiner

(73) Assignee: **IPAC Acquisition Subsidiary I, LLC**,
 Peterborough, NH (US)

Primary Examiner—Wendy R. Garber
Assistant Examiner—Jacqueline Wilson

(74) *Attorney, Agent, or Firm*—Sawyer Law Group LLP

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 32 days.

(57) **ABSTRACT**

(21) Appl. No.: **09/625,824**

(22) Filed: **Jul. 26, 2000**

(51) Int. Cl.⁷ **H04N 5/232**

(52) U.S. Cl. **348/211.3; 348/211.12; 348/14.04**

(58) **Field of Search** **348/14.01, 14.02, 348/14.04, 14.08, 14.09, 552, 143, 156, 157, 211.3, 211.12; 709/322**

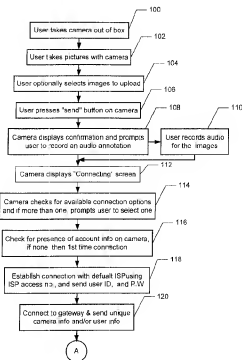
(56) **References Cited**

U.S. PATENT DOCUMENTS

5,430,827 A * 7/1995 Rissanen 704/272
 5,737,491 A * 4/1998 Allen et al. 704/270
 5,905,736 A * 5/1999 Ronen et al. 370/546
 6,064,671 A * 5/2000 Killian 370/389
 6,067,571 A * 5/2000 Igarashi et al. 709/232

The present invention provides a method and system for automatically establishing a user account and for configuring a hand-held electronic device for accessing a site on a public network using the user account. The method and system include establishing a connection to a website server, which is responsible for establishing and maintaining website accounts. The presence of account information on the electronic device camera is then checked, and if none are found, it is determined that this is a first-time connection and information uniquely identifying the electronic device is sent to the website server so that the server can set-up the account information based on the electronic device information. The server then sends user account information to the device, including an account ID and password. The user account information is then stored on electronic device for use the next time the electronic device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network.

22 Claims, 6 Drawing Sheets



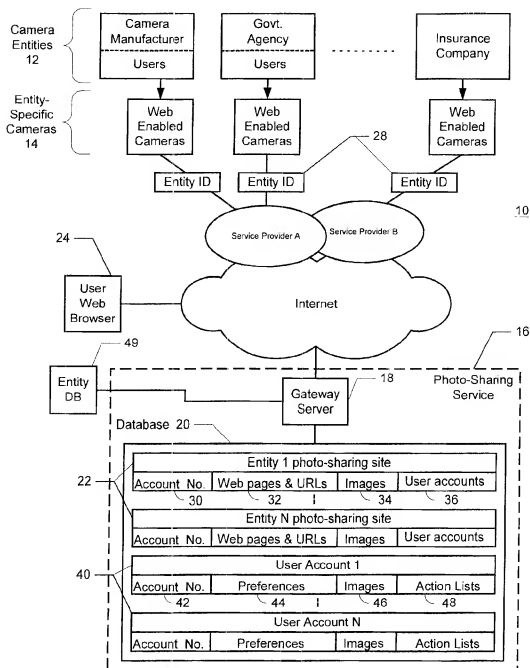


FIG. 1

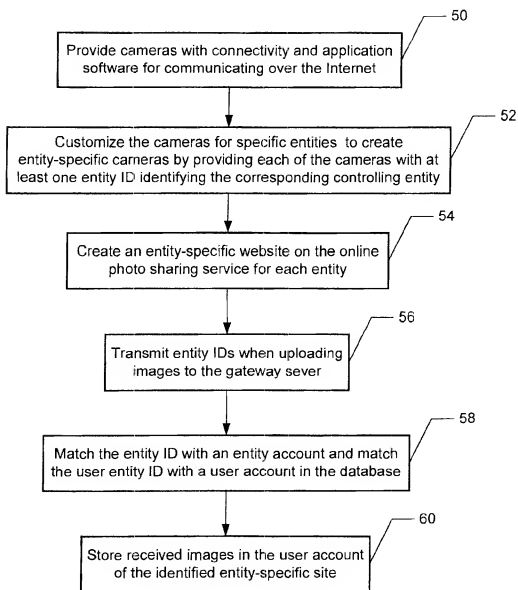


FIG. 2

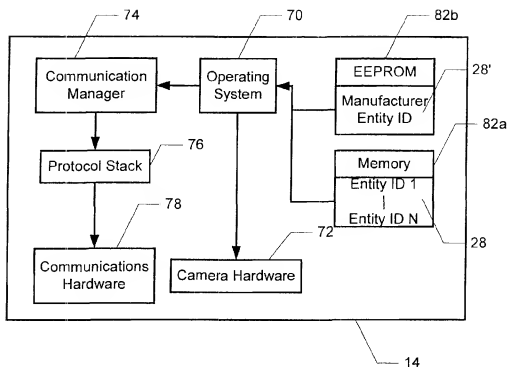


FIG. 3

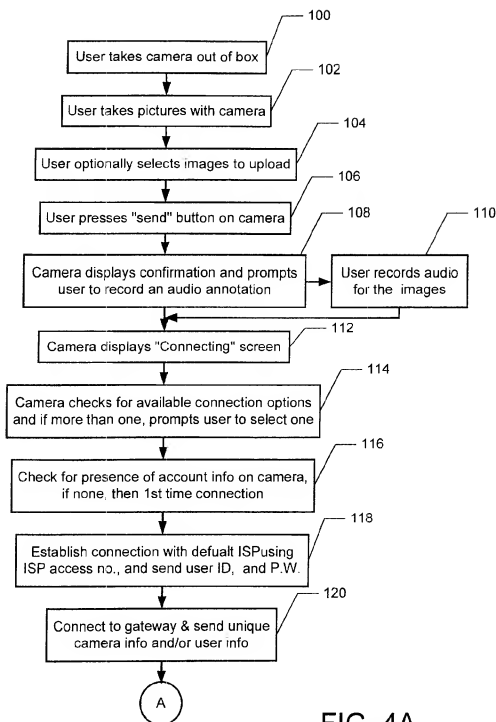


FIG. 4A

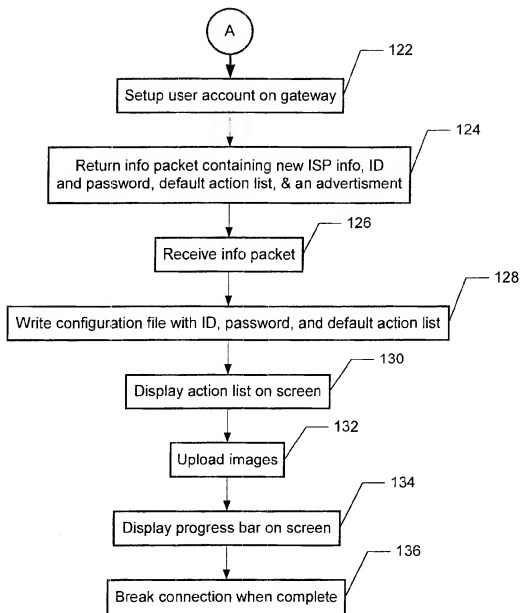


FIG. 4B

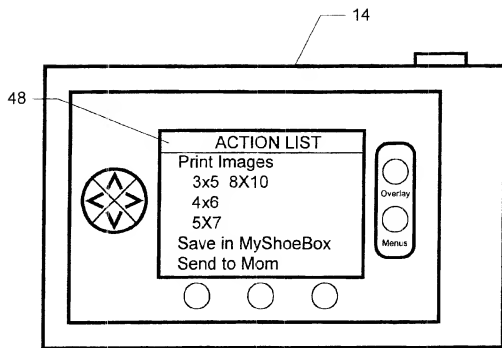


FIG. 5

1

AUTOMATICALLY CONFIGURING A WEB-ENABLED DIGITAL CAMERA TO ACCESS THE INTERNET

CROSS-REFERENCE TO RELATED APPLICATIONS

The present invention is related to co-pending U.S. patent application Ser. No. 09/625,398 entitled "Method and System For Hosting Entity-Specific Photo-Sharing Websites For Entity-Specific Digital Cameras"; and to co-pending U.S. patent application Ser. No. 09/626,418, entitled "Method And System For Selecting Actions To Be Taken By A Server When Uploading Images," which are assigned to the assignee of the present application and filed on the same date as the present application.

FIELD OF THE INVENTION

The present invention relates to a method and system for customizing digital cameras to upload images to an entity-specific photo-sharing websites, and more particularly to automatically configuring a web-enabled digital camera to access the Internet.

BACKGROUND

As the popularity of digital cameras grows, the desire of digital camera users to share their images with others will also continue to grow. New digital camera owners typically try to share their images based on the paradigm of film cameras, in which images are printed on paper and then placed into a photo album. The most straightforward approach to do this with a digital camera is to connect the digital camera directly to a printer to create the prints, and then manually insert the images into a photo album. Users often find this process somewhat complicated and restrictive because standard printers can only print images in limited sizes and requires particular types of paper. And even after the photo album has been assembled, the printed images are not easily shared with many people.

The best approaches to photo-sharing take advantage of the Internet. One such approach is for users to store the digital images on a PC and then send the images to others using email. Several Internet companies now offer an even more convenient approach by providing photo-sharing websites that allow users to store their images for free and to arrange the images into web-based photo albums. Once posted on a photo-sharing website, others may view the images over the Internet.

While convenient for storing digital images, getting the images to the photo-sharing websites can be challenging for users. Most commonly, users must upload their images from the digital camera to a PC using a cable or IrDA, or by inserting the camera's flash card into the PC. From the PC, the user logs onto the Internet and uploads the images to a photo-sharing website. After uploading the images, the user works on the website to arrange the images into web albums and to add any textual information.

Although today's approach for storing images from a digital camera onto a web photo-sharing website and for creating web photo albums works reasonably well, two problems remain that hinder the mainstream adoption of web-based photo-sharing. One problem is that this approach requires the use of a PC, notebook computer, or PDA. While many digital camera users today have PC's, most of those users are early adopters of technology. There are many other consumers who would purchase a digital camera, but are

2

reluctant to do so because they do not yet own a PC or are intimidated by them.

In an effort to address this problem, the assignee of the present application developed an approach to uploading images to the web that doesn't require the use of a PC. In this approach, an email software application is loaded into a digital camera capable of running software that allows the user to e-mail the images directly from the camera. The user simply connects his or her digital camera to a cellphone or modem, runs the e-mail application, and selects the desired images and the email recipients. The selected images are then sent to the recipients as e-mail attachments.

Although emailing photos directly from the camera allows users who do not own a PC to share images over the Internet, these users must still establish accounts with both an Internet service provider (ISP) and the photo-sharing website before being able to post their images. These accounts must also be set-up by PC users as well. For techno savvy users who use a PC to upload the images to the photo-sharing website, establishing the accounts may not be a bother, but even these users may not always have their PC's handy, such as when on vacation, for instance. And for non-PC users, establishing the accounts by entering account information on the digital camera itself may prove to be a cumbersome, if not a daunting, task.

Accordingly, what is needed is an improved method for uploading images from a digital camera to a photo-sharing website on the Internet. In order for online photo-sharing to become more mainstream, an approach that doesn't require a PC or PC expertise and that reduces complexity for the user is required. The present invention addresses such a need.

SUMMARY OF THE INVENTION

The present invention provides a method and system for automatically establishing a user account and for configuring a hand-held electronic device for accessing a site on a public network using the user account. The method and system include establishing a connection to a website server, which is responsible for establishing and maintaining website accounts. The presence of account information on the electronic device camera is then checked, and if none are found, it is determined that this is a first-time connection and information uniquely identifying the electronic device is sent to the website server so that the server can set-up the account information based on the electronic device information. The server then sends user account information to the device, including an account ID and password. The user account information is then stored on electronic device for use the next time the electronic device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network.

According to the method and system disclosed herein, a user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network with the electronic device.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG 1 is a block diagram illustrating an online photo-sharing system in accordance with a preferred embodiment of the present invention.

FIG 2 is a flow chart illustrating a business method for hosting entity-specific photo-sharing websites for entity-specific image capture devices in accordance with a preferred embodiment of the present invention.

3

FIG. 3 is a diagram showing a preferred embodiment of the connectivity and application software of the camera.

FIGS. 4A and 4B illustrate a flow chart of a process for automatically configuring a web-enabled digital camera to access the Internet in a preferred embodiment of the present invention.

FIG. 5 is a block diagram illustrating an example action list 48 displayed on the LCD screen of the camera.

DETAILED DESCRIPTION

The present invention relates to an automatic system for uploading images from a digital camera to entity-specific photo-sharing websites and for automatically establishing accounts. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

FIG. 1 is a block diagram illustrating an online photo-sharing system 10 in accordance with a preferred embodiment of the present invention. The system includes a plurality of camera controlling entities 12 that produce, own, or otherwise control a set of digital cameras 14, and an online photo-sharing service 16. The online photo-sharing service 16 includes a gateway server 18 and an entity/account database 20. The various camera controlling entities 12 contract with the photo-sharing service 16 to transparently host customized photo-sharing websites 22 for each entity, which are referred to herein as entity-specific photo-sharing websites 22. The entity-specific photo-sharing websites 22 are each accessible to the user through the entity's existing Internet site (not shown), and thus appear to users as though the entity-specific photo-sharing websites 22 are hosted by the corresponding entity. According to a preferred embodiment of the present invention, the cameras 14 for a particular entity are customized for that entity to create entity-specific cameras 14, such that when the cameras 14 connect to the Internet, the cameras 14 automatically upload their images to the photo-sharing website of the corresponding entity. In a further aspect of the present invention, the photo-sharing service 16 automatically stores the images in a web album, which is viewable over the Internet by a user's web browser 24.

As used herein, a camera controlling entity 12 is any entity that makes, owns, sells, or controls digital cameras 14, and therefore includes, camera manufacturers, companies, retailers, and end-users. One or more combination of these entities 12 may either contract with the photo-sharing service 16 to provide entity-specific websites 22 for their cameras 14, or have entity information transmitted to the photo-sharing service 16 from the cameras 14. Therefore, a camera controlling entity 12 may include a single entity 12 or a hierarchical relationship of entities 12.

An example of a single entity 12 includes an insurance company that contracts with the photo-sharing service 16 to have all digital cameras 14 used by their agents to transmit their images to a customized insurance photo-sharing website. Examples of a hierarchical relationships of entities 12 includes a camera manufacturer, such as Nikon, that contracts with the photo-sharing service 16 to have all Nikon digital cameras 14 transmit their images to the customized

4

Nikon photo-sharing website. Since the images of different users must be distinguished, each user of a Nikon camera 14 would also constitute an entity within the Nikon website so that the images from different users can be distinguished. Other examples of hierarchical entity relationships include a retailer and its consumers, a real estate agency and its agents, community groups and its members, and government agencies and its employees, for instance.

In a preferred embodiment, the cameras 14 are customized for their respective entities 12 by providing the cameras 14 with software for transmitting entity ID information 28 identifying its controlling entity 12 to the photo-sharing service 16. The photo-sharing service 16 in conjunction with the gateway server 18 and the entity/account database 20 hosts the entity-specific photo-sharing websites 22. Each entity-specific website 22 is identified in the database 20 by an entity account number 30 and includes the web pages and URIs 32 comprising the website, the images and web albums 34 stored on the website, and the user account numbers 36 of authorized users. The database 20 also includes user accounts 40, each of which comprises a user account number 42, user preferences 44, the user's images 46, and action lists 48, explained further below.

The gateway server 18, which communicates with the cameras 14 during image uploading, receives one or more entity IDs 28 from each camera 14 and matches the entity ID 28 with an entity account 30 in the database 20. The images are then automatically associated with the photo-sharing website 22 of the identified entity 12 and/or the identified user.

After the images are uploaded, a user of the camera 14 may visit the online photo-sharing website 22 over the Internet to view the images via a web browser 24. Since the photo-sharing websites 22 are transparently hosted by the photo-sharing service 16, each photo-sharing website 22 appears as though it is hosted by the entity itself, rather than the third party service.

In one embodiment, the cameras 14 may connect to the Internet via a service provider 26, which may include a wireless carrier and/or an Internet service provider (ISP) that is capable of servicing many devices simultaneously. In a preferred embodiment, each of the cameras 14 is provided with wireless connectivity for connecting to the Internet, and are therefore so called "web-enabled" devices, although a wired connection method may also be used.

The cameras 14 may be provided with wireless connectivity using anyone of a variety of methods. For example, a cellphone may be used to provide the digital camera 14 with wireless capability, where the camera 14 is connected to the cellphone via a cable or some short-range wireless communication, such as Bluetooth. Alternatively, the camera 14 could be provided with built-in cellphone-like wireless communication. In an alternative embodiment, the digital camera 14 is not wireless, but instead uses a modem for Internet connectivity. The modem could be external or internal. If external, the camera 14 could be coupled to modem via any of several communications means (e.g., USB, IEEE1394, infrared link, etc.). An internal modem could be implemented directly within the electronics of camera 14 (e.g., via a modem ASIC), or alternatively, as a software-only modem executing on a processor within camera. As such, it should be appreciated that, at the hardware connectivity level, the Internet connection can take several forms. Hence, it should be appreciated that the present invention is not limited to any particular method of accessing the Internet.

5

In a preferred embodiment, the entity-specific websites 22 are customized to seamlessly integrate into the entity's existing website by following the look and feel of the entity's existing website. The entity-specific websites 22 are hosted on the photo-sharing service 16, but a link to the entity-specific websites 22 may be provided on the homepage of the corresponding entity's existing website. Thus, in order to view a web album on an entity-specific website 22, the user must visit the entity's existing website and click the link to the entity-specific website 22, where the user's browser 24 will be transparently directed to the photo-sharing service 16 and be provided with the web pages 32 of the entity-specific website 22.

As an example of the operation of the photo sharing system 10, consider the following scenario. Assume that Minolta and Nikon are entities 12 that have contracted with the photo-sharing service 16, and that the photo-sharing service 16 hosts a photo-sharing website 22 for Minolta and a photo-sharing website 22 Nikon. The Minolta cameras 14 would be provided the entity ID 28 for Minolta and the Nikon cameras 14 would be provided the entity ID 28 for Nikon. When the Minolta and the Nikon cameras 14 send sets of images to the photo-sharing service 16, the gateway server 18 would distinguish the cameras 14 by the entity IDs 28 and would direct the set of images received from Minolta cameras 14 to Minolta's photo-sharing website, and would direct the images from Nikon cameras 14 to Nikon's photo-sharing website. To view the images, the owners of the cameras 14 would use a browser 24 on their PC or PDA to visit the URL of the Minolta or Nikon photo-sharing websites 22. In one preferred embodiment, the photo-sharing service 16 sends the URL of the entity-specific website 22 directly to the camera 14 for display to inform the user of the address.

According to the present invention, the photo-sharing service 16 provides business-to-business and business-to-consumer business models. The service is business-to-business because the service provides companies, such as camera manufacturers, with a complete end-to-end solution for their cameras 14. The solution includes customized software for their cameras 14 for sending images over the internet, and an internet website for storing images from those cameras 14 on a branded website that appears to be hosted by the company. The service is business-to-consumer because it allows users of digital cameras 14 with an automatic solution for uploading captured images from a digital camera 14 to an online photo-sharing website, without a PC.

According to one preferred embodiment of the present invention, the photo-sharing service 16 provides a method of doing business whereby the photo-sharing service 16 shares revenue based on the hierarchical relationship of the entities 12. For example, if the photo sharing service 16 charges a fee for receiving and/or storing the images received from the entity-specific cameras 14, then the photo sharing service 16 may share a portion of the fee with the manufacturer and/or third party supplier of the camera 14 that uploaded the images, for instance. Revenue may also be shared with the wireless service provider providing the connection with the photo sharing service 16.

FIG. 2 is a flow chart illustrating a business method for hosting entity-specific photo-sharing websites for entity-specific image capture devices, such a digital cameras, in accordance with a preferred embodiment of the present invention. First, the cameras 14 are provided with connectivity and application software for communicating over the Internet in step 50. In a preferred embodiment, this step is

6

performed during camera 14 manufacturing to provide off-the-shelf web enabled cameras 14. The cameras 14 are also customized for specific entities 12 to create entity-specific cameras 14 by providing each of the cameras 14 with at least one entity ID 28 identifying the corresponding controlling entity in step 52.

Referring now to FIG. 3, a diagram showing the preferred embodiment of the connectivity and application software of the camera 14 and the entity ID 28 information is shown. Preferably, the camera 14 includes a microprocessor-based architecture that runs an operating system 70 for controlling camera hardware 72 and overall functionality of the camera 14 (e.g., taking pictures, storing pictures, and the like). An example of such an operating system 70 is the Digita™ Operating Environment developed by the assignee of the present application. The camera 14 also includes communication manager 74 software, and a TCP/IP protocol stack 76, that enables communication via the internet, as is well-known in the art. The entity ID information 28 and captured images may be stored in one or more types of memories 82.

For hierarchical entity relationships, the cameras 14 are provided with hierarchical entity IDs 28; one entity ID 28 identifying the entity, and a second entity ID 28 identifying the end-user. Whether there are one or more entity IDs 28, the entity ID 28 of the camera manufacturer may always be provided. Camera 14 customization may occur either during manufacture or anytime thereafter. In a preferred embodiment, the manufacturer entity ID 28 is provided at the time of manufacturing and is stored in an EEPROM 82b, while the entity IDs 28 for other entities 12, such as companies and end-users, are loaded into the camera 14 subsequent to manufacturing and are stored in flash memory 82a or the EEPROM 82b.

Customization that occurs subsequent to manufacture may be implemented using several methods. The first method is to manufacture the cameras 14 with an application programming interface (not shown) for accepting a subsequently loaded software application that specifies the entity ID's 28. The application may come preloaded on a flash card, which is then inserted into the camera 14 by the user and stored in flash memory 82a. The application may also be wirelessly beamed into the camera. When executed in the camera, the software application transmits the appropriate entity IDs 28 to the gateway server 18.

The second method is to load a small file in the camera 14 specifying the entity IDs 28 from a removable memory or from a PC, and storing the file in a system folder within the camera's flash memory 82a. The camera 14 then accesses the file when an Internet connection is established. In a preferred embodiment, the communication manager 74 automatically extracts the manufacturing ID 28 and the entity ID 28 and transmits them to the gateway server 18. In this embodiment, the entity ID 28 is also stored in the EEPROM 82b and is factory set to zero (empty). Thus, unless the entity ID 28 is set, the manufacturing ID 28 may default as the highest controlling entity.

If, for example, a third party developer X contracts to provide custom camera software for camera manufacturer Z, then a custom entity ID will be issued for developer X and developer X will place the custom entity ID into the EEPROM 82b. Developer X is now a controlling entity 12, and may specify to the photo-sharing service 16 that a developer X entity-specific photo-sharing site 22 or developer X's own website be the destination for the uploaded images.

7

The protocol stack 76, under direction of the communications manager interfaces with the communications hardware 78 of camera. The protocol stack 76 includes software APIs and protocol libraries that interface with the communication manager 74, and communication hardware interface drivers that interfaces directly with the various communication hardware 72 the camera 14 must function with (e.g., a Bluetooth transceiver, etc.). The communication manager 74 communicates with operating system 70 and the IP protocol stack 76 to establish an Internet connection ant to transmit the entity ID 28 information and images from the memories 82a and 82b to the photo-sharing service 16.

In an alternative embodiment, rather than loading entity ID's 28 into the camera, a combination of the camera's serial number and the make and model number of the camera may be used as the entity ID 28. Entity specific cameras 14 may then be distinguished by providing a mapping of the camera serial numbers and product IDs to specific entities 12 in the database 20.

Although the camera 14 has been described in terms of a software-based customization solution, those with ordinary skill in the art will readily recognize that the camera 14 may also be provided with a hardware-based solution.

Referring again to FIG. 2, before or after camera customization, the entity-specific websites 22 are created for each entity contracting with the photo-sharing service 16 in step 54. Customization requires storage of entity information in the entity/account database 20 and creating and storing web page elements comprising the entity-specific photo-sharing website in the database 20. The entity-specific information stored in the database 20 may also include service levels, and enabled features for the entity-specific website 22. Features are components or services that may be provided on websites by the photo-sharing service 16, such as search functions, and online printing, for instance, but may be selectable by each entity for its own website. As an example, company X may provide customized cameras 14 for its employees, but may not wish to allow employees to print images from the company X photo website for security reasons. If so desired, company X may have the photo service disable this feature from their particular website.

In a preferred embodiment, the entity-specific websites 22 are not created from scratch, but are created by modifying a preexisting template. The template may include several different sections, such as A, B, C and D, for instance. Assuming for example that the template used to create a website for Nikon, and section A is used to specify the name of the entity then the name Nikon would be inserted into that section. Other entity-specific content would be used to fill out the remaining sections. The Web pages comprising the Nikon specific photo-sharing website would then be provided with URL's unique to that website. The entity's regular website would be modified to include a link to the entity's photo-sharing website 22. In addition, the entity-specific photo-sharing website would include a link back to the entity's website. Entities 12 may have entity photo-sharing websites 22 created for them in one of two ways; automatically by logging into the photo-sharing service 16 and manually customizing the templates, or by having the entity photo-sharing website created for them.

Referring still to FIG. 2, when a camera 14 establishes an internet connection with the gateway server 18, the camera 14 transmits its entity IDs 28 and/or user entity ID 28 when uploading user selected images to the gateway server 18 in step 56. In response, the gateway server 18 matches the entity ID 28 with an entity account in the database 20 and

8

matches the user entity ID 28 with a user account 40 in the database 20 in step 58. The images received are then stored in the user account 40 of the identified entity-specific website 22 in step 60.

Referring again to FIG. 1, each user account 40 in the database 20 may also include one or more action lists 48. According to the present invention, an action list 48 includes one or more items representing actions that the gateway server 18 should take with respect to uploaded images, such as where to store and/or send the images from a particular user or camera, for instance. As explained further below, the action list 48 stored on the database 20 under a user's account 40 are automatically downloaded to the user's camera 14 during a connection with the gateway server 18 and stored on the camera 14. When the user initiates an image upload, the action list 48 is displayed to the user so the user may easily select what actions the gateway server 18 should take with respect to the images by selecting the displayed action list items.

Examples of action list items include specifying that the uploaded images should be stored on the entity-specific photo website, sending the images to a list of email addresses, or even performing some type of analysis or calculation on the image data, for instance.

In a further aspect of the present invention, an action list item is not limited to, instructing the gateway server 18 to perform actions only within the photo-sharing service 16. Rather, an item in the action list 48 may also instruct the gateway server 18 to perform actions outside of the photo-sharing service 16, such as storing the images in an external database 49 of the entity 12. For instance, in the example where the entity 12 is a company, some users of the company's cameras 14 could have action lists 48 instructing the gateway server 18 to store uploaded images to the company's database, rather than to the company's photo-sharing site 22. Based on the action lists 48 and customization, the gateway server 18 may be programmed to automatically perform predefined tasks, such as creating new web albums, or a new page within an existing album, parse the images to extract sound files or other metadata, print images and mail them to designated addresses, and so on.

In a preferred embodiment, the action lists 48 may be created via several methods. In one method, the action list is created by the photo-sharing service 16 the first time the user's camera establishes a connection. That is, a default action list 48 is automatically created based on the entity ID when a user account 40 is first created. In a hierarchical entity relationship where the entity 12 is a company, a default action list 48 may be created to implement a workflow specified by the entity 12. In a hierarchical relationship where the entity 12 is camera manufacturer, for instance, a default action list 48 may be created instructing the gateway server 18 to store the user's images in a simulated "shoebox" on the entity-specific photo-sharing site 20. The user may then go online and create albums from the images in the shoebox as desired.

Another method is for the user to create the action list 48 online on the entity-specific photo-sharing site 20. The action list 48 may be created manually on the website 20 by the user navigating to the site 20 using a web browser 24, accessing her account, and manually creating the action list 48 or editing the action list 48 on the entity-specific site 22. The action list 48 may also be created automatically on the website 20 in response to user actions performed on the website, such as printing images, or creating a web album.

9

Alternatively, after performing an action, the user may be prompted whether they would like this action added to his or her action list 48. If so, the user clicks a check-box and the item is added the action list 48. In a preferred embodiment, any action list 48 created and edited on the photo-sharing site 20 are downloaded to the camera every time the camera 14 connects to the photo-sharing service 16 and made available for user selection on the camera 14 during the next upload.

Yet another method for creating an action list 48 is to allow the user to create the action list on the camera 14. The user may manually create an action list 48 by "typing" in predefined items on the camera. The user may also type in an email address as an action list item whereby when that item is selected, the uploaded images are stored as a web album on the entity-specific photo-sharing website 22 and the server 18 sends a notification to the specified recipient containing the URL to the web album page.

A method and system for hosting web-based photo-sharing websites and for customizing digital cameras to upload images to the entity-specific photo-sharing websites has been disclosed. According to the present invention, users of the customized cameras 14 can upload images to the Internet for storage and web photo album creation without the use of a PC.

In one embodiment described above, the present invention assumes that an ISP account has been established with the digital camera's service provider, and that users of cameras 14 belonging to a certain entity 12 may use the cameras 14 to upload images to the website of the entity 12. However, two problems with account setup remain. One problem is that just as with a PC and PDA, the user must first establish an ISP account before the camera 14 can establish Internet communication. The second account problem is that most websites, including the photo sharing sites 22, may require each user to establish a unique account before using the site to distinguish one user from another. Before being able to connect the web-enabled camera 14 to the Internet, the user must establish these two accounts by either entering account setup information on a PC or entering account setup information on the camera 14. Neither alternative is a convenient alternative for people who do not have the time nor inclination to do so.

In a further aspect of the present invention, the cameras and the photo sharing site are provided with software for automatically creating Internet and photo-sharing website accounts for each camera 14 upon first use, without requiring the user to first enter account information on a PC or on the camera 14.

Referring now to FIGS. 4A and 4B, a flow chart illustrating a process for automatically configuring a web-enabled digital camera to access the Internet in a preferred embodiment of the present invention. Although the process will be described in terms of automatically establishing Internet accounts for a digital camera without requiring the user to enter information, those with ordinary skill in the art will readily recognize that the present invention may be used to automatically establish Internet accounts for any type of portable electronic device.

The process assumes that a user has just acquired a digital camera 14 customized as described above, and has just taken the camera 14 out of its box in step 100. After taking pictures with the camera in step 102, the user may review the images in the camera's LCD screen and optionally select a set of images to upload to the photo sharing service 16 in step 104. The user then presses a "send" button on the camera in step 106 to upload the images.

10

In response, the camera displays a confirmation dialog screen on the camera and prompts the user to record an audio annotation for the images or to continue in step 108. The user may then choose to record audio for the images in step 110. After choosing to continue, or after recording audio, the camera displays a "connecting" dialog screen in step 112 to indicate to the user that the camera is establishing an Internet connection. At the same time, the camera checks for available connection options in step 114, and if more than one is found, the camera prompts the user to select one of the connection options. For example, the camera may be within range of a Bluetooth-equipped printer and a cellphone, so the user will be prompted to choose which device the camera should establish communication with.

The camera then checks for the presence of account information on the camera in step 116, and if there are none, the camera assumes that this is a first-time connection. According to the present invention, in order to allow the camera to make a first-time Internet connection, the camera is provided with default Internet service provider (ISP) information during manufacturing, including an ISP access number, and user ID and password (if required). The camera establishes connection with the default ISP in step 118 by dialing the preloaded access number, and by sending the preloaded user ID and password to the ISP. This special account may be configured so that the camera can only connect to the gateway server 18 (no other IP addresses may be allowed).

After connecting with the ISP, the camera connects to the gateway server 18 and sends unique camera information and/or user information in step 120. In a preferred embodiment of the present invention, a combination of the camera's serial number and the make and model number of the camera may be sent as the unique camera information. In another preferred embodiment, the user's e-mail address may be sent as the unique camera or user information.

Continuing with FIG. 4B, the gateway server 18 uses the unique camera information to set up a user account 40 in step 122. After creating the user account 40, the gateway server 18 returns an information packet to the camera containing new ISP information (if needed), an account ID, and an account password in step 124. The information packet may also contain a default action list specifying what actions should be taken with respect to the images, an advertisement for display on the camera, and the URL of the entity-specific website 22.

It should be noted that if the camera is used in conjunction with an IP direct phone or is provided with a phone number for connected to a dedicated server where the user is not billed separately for the ISP connection, then the steps of providing the camera with default ISP info and returning new ISP info, may be omitted.

The camera receives the information packet in step 126, and writes a configuration file to memory 82a containing the ID, password, and default action list in step 128. The camera then displays the action list on the camera's LCD screen for selection by the user in step 130. The camera may optionally display the user's account information as well.

In an alternative preferred embodiment where security is a concern, the camera 14 first logs off the special ISP and the gateway accounts, and then reconnects using new ISP and gateway accounts in order to retrieve the action lists 48.

FIG. 5 is a block diagram illustrating an example action list 48 displayed on the LCD screen of the camera 14. The action list 48 is shown displaying three major options; printing the uploaded images, saving the uploaded images in

11

the user's shoebox, and sending the images to Mom. Under the printing option, the user may select from various size prints. Rather than nested menu categories as shown under the printing option, the action list 48 may be displayed with each action listed as a separate item (e.g., "Send 4x8 prints to Mom", "Send 5x7 prints to me").

Referring again to FIG. 4B, after the user selects one or more actions from the action list 48, the camera begins to upload the images along with the selected actions in step 132 and displays a progress bar on the screen in step 134. In one preferred embodiment, the camera may also display the advertisement sent in the information packet from the gateway server 18. The advertisement may advertise the controlling entity 12, the entity's photo-sharing site 22, or the photo sharing service 16. After all the images are uploaded and associated with the user's account 40, the camera breaks the connection with the gateway server in step 136. At this point, the camera 14 may also display the URL of the entity-specific website 22 to the user.

The next time the user uploads images, the camera will use any new ISP information received to connect to the Internet and will use the account ID and password written to memory 82a when connecting to the gateway server 18. Thus, by using unique camera information, such as the serial number, to establish a web site account upon first use, the present invention eliminates the need for the user to type in information to establish a web site accounts.

To further explain the present invention from a user interaction point of view, consider the following scenario where a user named Jack has just purchased a digital camera 14 from a store and unpacks the camera from the box. There is a "Quick Start Guide" which guides him in getting started. Jack pops in the batteries, sets the date and time, and takes some pictures of his dog and his new baby. Jack can see the pictures have come out well on the small LCD, and now wants to try sharing them with his parents.

The Quick Start Guide says to select the photos to be sent using the "Select" button, and then press the "Send" button. So, Jack navigates to each of the baby pictures, selects them, and then presses the "Send" button. Instantly, a dialog comes up on the LCD screen: "No Receiving Device Found! Please turn on your phone or other connecting device." Oops! Jack pulls out his cell phone, and turns it on. He presses the "Continue" button. Jack does not see this happen, but the camera now "discovers" the cell phone, and immediately presents another dialog: "4 Images Selected. Press Record to add a Sound Note, or Continue to send". Although Jack finds the proposition of recording sound intriguing, he decides to skip it and presses the "Continue" button. Immediately, the dialog is replaced with a "Connecting . . ." dialog.

Shortly, another dialog appears: "Your camera serial number is 38147. Please write this down. You will need it to access your web photo albums". Jack writes the number down on the spot provided in the Quick Start Guide, and presses OK.

Another "Connecting . . ." dialog appears, and is then quickly replaced by another dialog, which says "A free, temporary account has been set up for you at www.photo-sharing.service.com/new_accounts. You will need your camera serial number to access your photos and complete the setup of your account. Please complete the account setup within 30 days". Jack writes down the URL in the space provided in the Quick Start Guide, and presses the OK button. Jack doesn't know it, but during this dialog, the camera has begun transmitting his images and is already partially complete.

12

A new dialog comes up, with a progress bar. Jack is surprised to see that the transmission is already almost ½ done. Below the progress bar, the dialog says "Press 'Continue' to use camera during photo transmission, or wait for progress bar to complete". Jack is interested in watching how fast his images are transmitted, and decides to watch the progress bar complete. A "Transmission Successful" dialog appears. Jack presses the OK button. The camera returns to the review mode.

Jack is pretty excited—he just sent four baby pictures to the Internet. Jack then decides to see what happened to his images so he turns on his PC. After connecting to the Internet, Jack types in the URL from the Quick Start Guide. A Photo-sharing service web page appears, welcoming Jack to the photo-sharing service. After looking briefly at the welcome page, Jack types in his serial number from the Quick Start Guide, and selects his camera's model number from a pop-up menu. Jack clicks on a "Submit" button on the web page.

Jack now sees a page which shows thumbnails of the baby pictures he just sent, the page is entitled "My Shoebox". The page explains to Jack that he is looking at his on-line digital photo shoebox. Since the server 18 knows this is Jack's first visit, special help messages may appear. Various options are provided via buttons and text links. One that catches Jack's eye is CREATE WEB PHOTO ALBUM. Jack clicks this button, and works his way through the process of setting up an on-line photo album. This includes selecting photos from the shoebox, as well as selecting layout and style. One of the check-box items Jack is offered is "Make this album a camera Action List Item". Jack doesn't know what that is, so he clicks the Action List link, which brings up a brief description "If you check this box, you will be able to send pictures directly from the camera to this photo album!" Jack finds this interesting, so he closes the description window, and checks the box. Jack also enters the email address for his parents and his wife's parents, so they can be notified to come and see his photo album, which he has entitled "Our First Baby".

One of the buttons Jack does not click is the "Complete Account Setup" button. He knows that he has 30 days to do that chore, and figures he will get back to it later.

Jack's wife arrives home from shopping at this point, and Jack wants to show her how the new camera works. He starts by showing her the baby album on the PC, then decides to take some pictures of them holding the baby. Jack then selects the images, and presses the "Send" button again.

Since his cell phone is still on, sitting on the table a few feet away, the connection goes smoothly and quickly. A new dialog pops up, surprising Jack. It says "Select the destination for your pictures" and offers two choices: "My Shoebox" and "Our First Baby". Jack is amazed, he doesn't realize that the server 18 downloaded his action list to his camera during the connection. Jack decides to select the "Our First Baby" web album as the destination for the images, and clicks OK. The pictures are sent as before. After the transmission has completed, Jack goes to the PC to check on the photo album. When Jack refreshes the album page, he now sees the additional pictures he just sent, along with the pictures he sent before.

Jack spots a "Send Prints" link on the web page, and clicks it. He is led through a selection of print types, mailing addresses, and credit card info to make it possible to send prints. He is offered the option of completing the setup of his account. Jack decides to do that now, and proceeds to fill out the requested information, including his credit card number.

13

Once the account setup is complete, Jack continues the print order. One of the radio button items is "Make prints a separate Action List Item" or "Make prints part of your Action List". Jack remembers something about Action Lists from before, but is not sure what this means. The description says "Making a separate action list for prints allows you to decide in the camera to send to the photo album and send prints or to just send to the photo album." Jack thinks this is cool, and clicks the "Make prints a separate Action List" item. The next time Jack sends photos from the camera, his action list will be updated to present three choices: My Shoebox, Our First Baby, and Our First Baby w/Prints.

The underlying technology supporting this scenario are summarized below. Other functions and features are assumed, but not required for this scenario:

1. Two-way connection between camera and portal
2. Camera metadata included in the request
3. If not using an IP direct connection,
 - 3.1. Default ISP connection info built into the camera for the first connection (country specific)
 - 3.2. Downloaded assigned ISP information from the portal to the camera
4. Software capable of recognizing automatically a set of supported phones and adjusting the protocol to match
5. Action Lists maintained on the server that are automatically downloaded to the camera to update the camera selection list each time a connection is made
6. An On-line shoebox
7. Ability to download files to the camera from the server. The files could be text, GIF, animated GIF, JPG, or even a script or applet. This feature enables the ability to display advertisements on the camera, remind the user of remaining time to complete account setup, make special offers, and indicate limits reached.

A method and system for automatically configuring a web-enabled digital camera to access the Internet has been disclosed. Although the present invention has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations to the embodiments and those variations would be within the spirit and scope of the present invention. For example, although the photo-sharing service has been described as including the gateway server and the database, the database may be located elsewhere. Also, the gateway server may be used to control account information, while one or more other servers may be used to provide the web pages of the entity-specific websites. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.

What is claimed is:

1. A method for automatically establishing a user account and for configuring a hand-held image capture device which includes a digital camera for accessing a site on a public network using the user account, the method comprising the steps of:

- (a) establishing a connection to a website server, the website server being responsible for establishing and maintaining website accounts;
- (b) checking for the presence of account information on the image Capture Device and if none are found, determining that this is a first-time connection and automatically sending image capture device information uniquely identifying the electronic device to the website server so that the server can set-up the account information based on the image capture device information;

14

(c) receiving user account information from the server, wherein the user account information includes an account ID and password; and

(d) storing the user account information on image capture device for use a next time the image capture device accesses the website, whereby the user does not have to enter account information in order to establish the connection with the public network or to establish the website account.

2. The method of claim 1 further including the step of providing a digital camera as the hand-held image capture device.

3. The method of claim 2 further including the step of sending at least a partial serial number as the image capture device information uniquely identifying the electronic device.

4. The method of claim 3 further including the step of uploading captured images to the website server.

5. The method of claim 4 wherein step (a) further including the steps of:

- (i) providing the image capture device with default ISP information; and
- (ii) establishing a connection to an ISP using the default ISP information.

6. The method of claim 5 further including the step of terminating the connection with the ISP when the uploading of the images is complete.

7. The method of claim 6 further including the step of providing the digital camera with a default ISP access number, user ID, and password.

8. A system for automatically establishing a user account and for configuring a hand-held image capture device which includes a digital camera image capture device for accessing a site on a public network using the user account, comprising:

means for establishing a connection to a website server, the website server being responsible for establishing and maintaining website accounts;

means for checking for the presence of account information on the image capture device, and if none are found, determining that this is a first-time connection and automatically sending image capture device information uniquely identifying the image capture device to the website server so that the server can set-up the account information based on the image capture device information; and

means for receiving user account information from the server, wherein the user account information includes an account ID; and

means for storing the user account information on image capture device for use a next time the image capture device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account.

9. The system of claim 8 wherein the hand-held image capture device comprises a digital camera.

10. The system of claim 9 wherein the image capture device information uniquely identifying the image capture device includes a serial number of the image capture device.

11. The system of claim 10 wherein the camera uploads captured images to the website server.

12. The system of claim 9 wherein the camera further includes means for providing the image capture device with default ISP information, and means for establishing a connection to an ISP using the default ISP information.

13. The system of claim 12 wherein the digital camera terminates the connection with the ISP upon completion of image uploading.

15

14. The system of claim 13 wherein the camera is provided with a default ISP access number, user ID, and password.

15. A method for automatically establishing a user account and for configuring a digital camera to access a website on a public network using the user account, the digital camera including captured images and communication means for accessing the public network, the method comprising the steps of:

- (e) storing default ISP information on the digital camera, the default ISP information including an access number;
- (f) in response to a user request to upload images to the website, determining whether this is a first-time connection;
- (g) if it is first-time connection, establishing communication with the default ISP by dialing the access number;
- (h) establishing communication with the website and automatically sending a digital camera ID to the website;
- (i) using the digital camera ID to set up a user account on the website;
- (j) sending user account information to the digital camera, the user account including an account ID and an account password;
- (k) storing the user account information on the camera; and
- (l) uploading images to the website and associating the images with the user account, whereby the user does not have to enter information to establish a connection with the public network or to establish the website account.

16. The method of claim 15 further including the step of sending at least a partial serial number as the digital camera ID.

17. The method of claim 16 further including the step of receiving new ISP information from the website.

18. The method of claim 17 further including the step of using the new ISP information a next time the digital camera establishes a connection with the public network.

16

19. A system for automatically establishing a user account and for configuring a digital camera for accessing a website on a public network using the user account, the digital camera including images captured by user and communication means for wirelessly accessing the public network, the system comprising the steps of:

- means for storing default ISP information on the digital camera, the default ISP information including an access number;
- means for determining whether this is a first-time connection in response to a user request to upload images to the website;
- means for establishing communication with the default ISP by dialing the access number;
- means for establishing communication with the website and sending a digital camera ID to the website;
- means for using the digital camera ID to set up a user account on the website;
- means for automatically sending user account information to the digital camera, the user account including an account ID and an account password;
- means for storing the user account information on the camera; and
- means for uploading images to the website and associating the images with the user account, whereby the user does not have to enter information to establish a connection with the public network or to establish the website account.

20. The system of claim 19 wherein the digital camera ID includes a serial number.

21. The system of claim 20 wherein the digital camera receives new ISP information from the website.

22. The system of claim 21 wherein the digital camera uses the new ISP information a next time the digital camera establishes a connection with the public network.

* * * * *



US00635384B1

(12) **United States Patent**
Morris

(10) **Patent No.:** **US 6,353,848 B1**
(45) **Date of Patent:** **Mar. 5, 2002**

(54) **METHOD AND SYSTEM ALLOWING A CLIENT COMPUTER TO ACCESS A PORTABLE DIGITAL IMAGE CAPTURE UNIT OVER A NETWORK**

(75) Inventor: **Robert P. Morris**, Raleigh, NC (US)

(73) Assignee: **FlashPoint Technology, Inc.**,
Peterborough, NH (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/127,514**

(22) Filed: **Jul. 31, 1998**

(51) Int. Cl.⁷ **G06F 15/16**

(52) U.S. Cl. **709/203; 709/200; 709/202; 709/217; 709/219; 709/238; 709/232; 709/245**

(58) **Field of Search** **709/200-203, 709/206, 217-219, 227-229, 231-232, 236-237, 245; 707/10, 200-204; 713/201-202**

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,911,044 A	~	6/1999	Lo et al.	709/203
6,018,774 A	~	1/2000	Mayle et al.	709/203
6,058,428 A	~	5/2000	Wang et al.	709/232
6,085,249 A	~	7/2000	Wang et al.	709/229
6,101,536 A	~	8/2000	Kolani et al.	709/217
6,141,759 A	~	10/2000	Braddy	709/203

FOREIGN PATENT DOCUMENTS

DE 198 08 616 9/1998 H04L/12/16

OTHER PUBLICATIONS

De Albuquerque et al., "Remote Monitoring Over The Internet", Nuclear Instruments & Methods In Physics Research, Section-A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 412, No. 1, Jul. 21, 1998, pp. 140-145.

* cited by examiner

Primary Examiner—Zarni Maung

Assistant Examiner—Bharat Barot

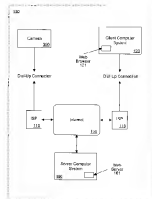
(74) Attorney, Agent, or Firm—Sawyer Law Group LLP

(57)

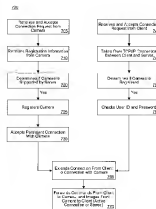
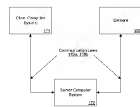
ABSTRACT

A method for accessing a digital image capture unit via a communication network comprising a server computer system and a client computer system communicatively coupled with communication equipment. In one embodiment, the address of the digital image capture unit is registered in an executable program on the server computer system. The executable program is accessed by the client computer system. The executable program connects the digital image capture unit and the server computer system. The executable program communicates commands between the client computer system and the digital image capture unit, such that data captured by the digital image capture unit is transferred to the client computer system via the server computer system.

27 Claims, 11 Drawing Sheets



126



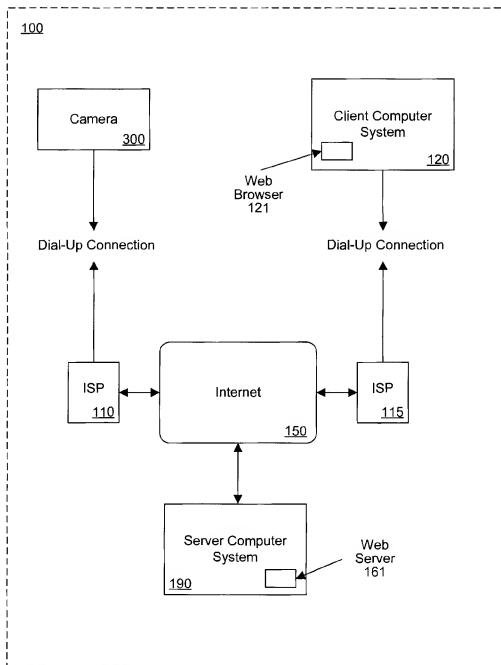


FIG. 1A

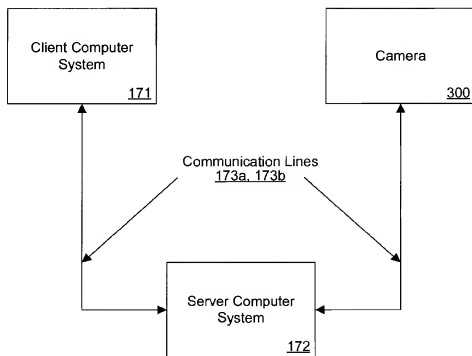
170

FIG. 1B

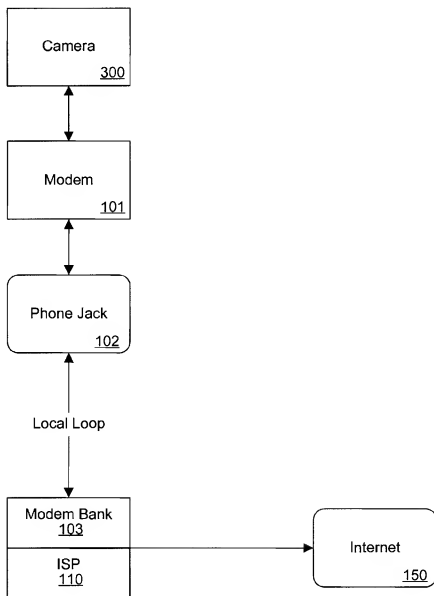


FIG. 1C

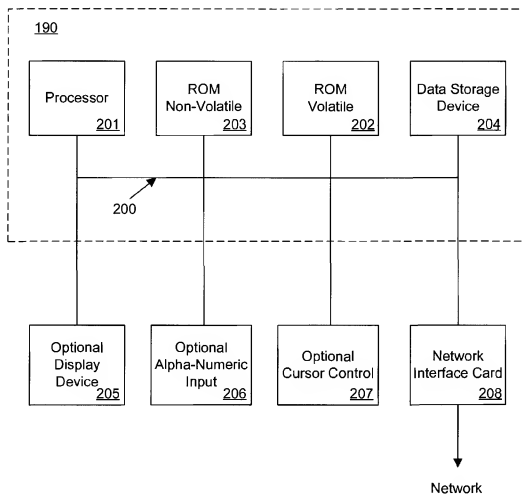


FIG. 2

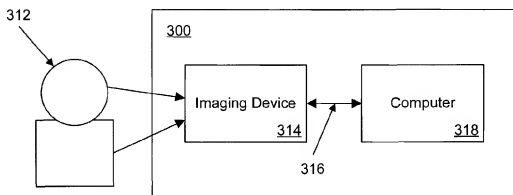


FIG. 3

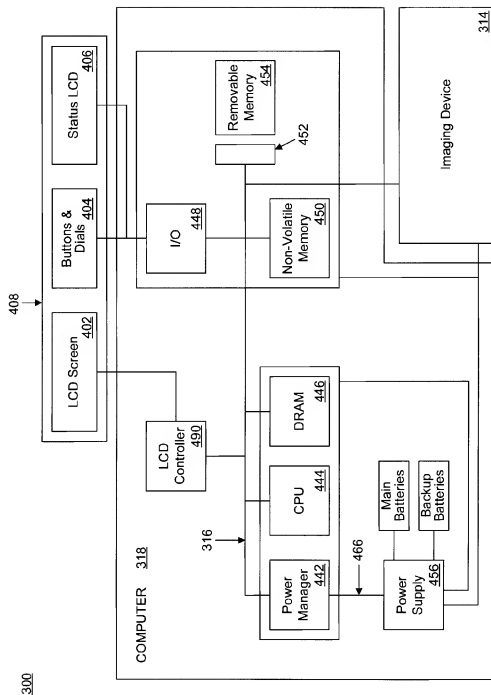


FIG. 4

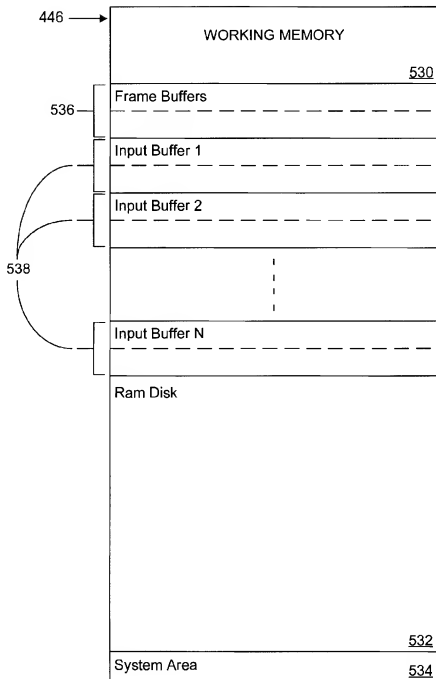


FIG. 5

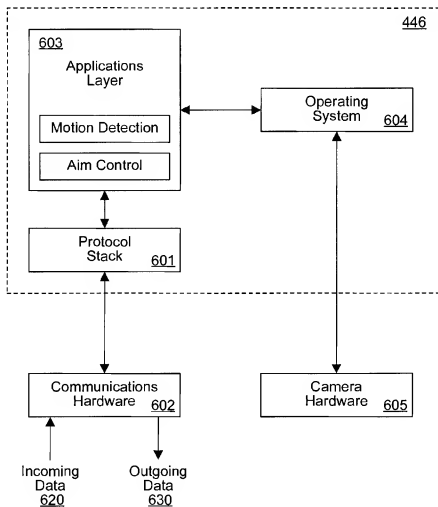


FIG. 6

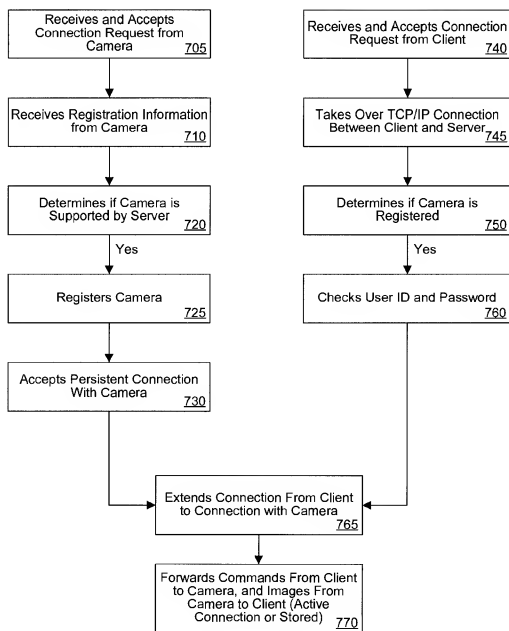
700

FIG. 7

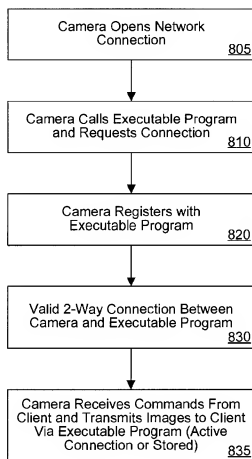
800

FIG. 8

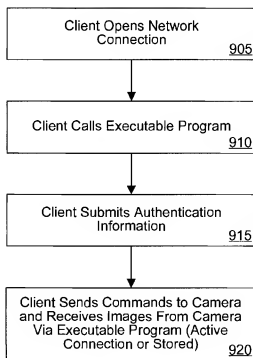
900

FIG. 9

METHOD AND SYSTEM ALLOWING A CLIENT COMPUTER TO ACCESS A PORTABLE DIGITAL IMAGE CAPTURE UNIT OVER A NETWORK

FIELD OF THE INVENTION

The field of the present invention pertains to digital image capture devices. More particularly, the present invention relates to a method for remotely accessing a digital camera via a communication network.

BACKGROUND OF THE INVENTION

Modern digital cameras typically include an imaging device which is controlled by a computer system running a software program. When an image is captured, the imaging device is exposed to light and generates raw image data representing the image. The raw image data are typically stored in an image buffer, where they are processed and compressed by the computer system's processor. Many types of compression schemes can be used to compress the image data, such as the joint photographic expert group (JPEG) standard. After the processor processes and compresses the raw image data into image files, the processor stores the image files in an internal memory or on an external memory card.

Some digital cameras are also equipped with a liquid-crystal display (LCD) or another type of display screen on the back of the camera. Through the use of the LCD, the processor can cause the digital camera to operate in one of two modes, play and record, although some cameras only have a record mode. In the play mode, the LCD is used as a playback screen allowing the user to review previously captured images either individually or in arrays of four, nine, or sixteen images. In the record mode, the LCD is used as a viewfinder through which the user may view an object or scene before taking a picture.

Besides the LCD, user interfaces for digital cameras also include a number of buttons or switches for setting the camera into one of the two modes and for navigating between images in play mode. For example, most digital cameras include two buttons, labeled "-", and "+", that enable a user to navigate or scroll through captured images. For example, if the user is reviewing images individually, then pressing one of navigation buttons causes the currently displayed image to be replaced by the next image.

A digital camera has no film and, as such, there is no incremental cost of taking and storing pictures. Hence, it is possible to take an unlimited number of pictures, wherein the most recent picture replaces the earliest picture, for virtually zero incremental cost. Accordingly, this advantage is best realized when the camera is used as much as possible, taking pictures of practically anything of interest.

One way to best utilize this unique attribute is to make the digital camera and its internally stored images remotely accessible. If the pictures are remotely accessible, the camera could be set to continuously take pictures of scenes and items of interest. Ideally, a user would be able to access those pictures at any time. The user would be able to use a widely available communications medium to access the camera from virtually an unlimited number of locations.

The emergence of the Internet as a distributed, widely accessible communications medium provides a convenient avenue for implementing remote accessibility. Providing remote accessibility via the Internet leverages the fact that the Internet is becoming familiar to an increasing number of

people. Many users have become accustomed to retrieving information from remotely located systems via the Internet. There are many and varied applications which presently use the Internet to provide remote access or remote connectivity. Internet telephony is one such application, such as Microsoft's NetMeeting and Netscape's CoolTalk.

NetMeeting and CoolTalk are both real-time desktop audio conferencing and data collaboration software applications specifically designed to use the Internet as their communications medium. Both software applications allow a "local" user to place a "call" to a "remote" user located anywhere in the world. With both NetMeeting and CoolTalk, the software application is hosted on a personal computer system at the user's location and on a personal computer system at the remote user's location. Both NetMeeting and CoolTalk require a SLIP (Serial Line Internet Protocol) or PPP (Point-to-Point Protocol) account where Internet access is via a dial-up modem, and where the user, as is typical, accesses the Internet through an ISP (Internet Service Provider). Both NetMeeting and CoolTalk require personal computer systems for the resources necessary to run these applications (e.g., processing power, memory, communications hardware, and the like). In addition, both NetMeeting and CoolTalk require the one user to input an IP (Internet Protocol) address for the other user in order to establish communication between the users.

To facilitate the process of obtaining appropriate Internet addresses, CoolTalk, for example, allows on-line users to list their respective IP addresses with a proprietary CoolTalk central Web server. This allows a user to obtain a list of users currently on-line to whom communication can be established. Upon locating the desired remote user in the Internet address list maintained by the Web server, the local user places the call.

In this manner, the proprietary CoolTalk Web server maintains a user-viewable and user-updated "address book" in which users list their respective Internet addresses and in which they search for the Internet addresses of others with whom they wish to communicate. However, both NetMeeting and CoolTalk require active user input, in that each require the user to input his current Internet address, and in that each require the local user to search the address book for the Internet address of the remote user to be contacted. This can be quite problematic in the case where users obtain access to the Internet via dial-up connections and hence have different Internet addresses each time their respective dial-up connections are established.

In a manner similar to Internet telephony, Internet desktop video conferencing is another prior art application which uses the Internet as its communications medium. One such application, for example, is CU-SeeMe by White Pine. CU-SeeMe provides real time video conferencing between two or more users. As with NetMeeting and CoolTalk, CU-SeeMe is a software application which runs on both the local user's personal computer system and the remote user's personal computer system. The personal computer systems provide the resources for running the application. As with NetMeeting and CoolTalk, CU-SeeMe requires the local user to enter the IP address of the remote user. Like CoolTalk, CU-SeeMe facilitates this process by allowing on-line users to list their respective IP addresses with a proprietary central Web server such that the addresses can be easily indexed and searched.

Another prior art example of remote access via the Internet is status queries of remote devices using the Internet as the communications medium. A typical prior art applica-

3

tion involves interfacing a remote device with a computer system, and accessing the computer system via the Internet. For example, a vending machine can be remotely accessed to determine its status (e.g., the number of sales made, whether the machine needs refills, whether the machine needs maintenance, and the like). The machine is appropriately equipped with sensors, switches, and the like, which in turn are interfaced to a computer system using a software driver. The computer system is coupled to the Internet and interfaces with the machine through the driver, making the relevant information available over the Internet using Web server software. Hence, any interested user (e.g., the vending machine service company) is able to remotely ascertain the status of the machine via the Internet.

A problem with the above described prior art applications is that access to the Internet and communication thereon require a separate host computer system (e.g., a personal computer system) on each side of the Internet connection in addition to the server computer system on the Internet. The two host computer systems provide the computational resources to host the respective software applications, the Internet access software, and any necessary device drivers. The required computational resources consume a significant amount of memory. Because of this, among other reasons, the above prior art applications are not easily transferred to the realm of easy-to-use, intuitive, consumer electronic devices such as digital cameras, which are small in size and so generally constrained by the amount of memory they can house. In addition, a consumer electronic device such as a digital camera that requires a separate computer system would be more expensive and complex, and therefore would not be consistent with the desire of consumers for lower cost and simpler devices.

Also, separate host computer systems (where the host computer systems host the software and drivers required by prior art applications as described above) require extra effort to administer, particularly with regard to networks consisting of a large number of computer systems (e.g., digital cameras each incorporating a computer system). For example, an upgrade to the software residing on each computer system has to be individually installed on each computer system. Also, each computer system has to be individually polled to query whether the computer system has data of interest to the user, and then the data have to be separately accessed and collected from each computer system, then compiled. For example, in an application involving digital cameras, a user may be interested in finding out which digital cameras have images in storage. In the prior art, the user has to access each digital camera individually. In another case, a user may have an interest in maintaining a record of transactions between all users and all digital cameras. Again, in the prior art this is accomplished by individually accessing each digital camera (or, alternatively, each user's computer system) to collect the data, and then compiling the complete list of transactions.

Another problem with the prior art is the fact that the applications described above require the user to know the Internet address of the person or device that is being contacted. The Internet telephony applications (e.g., CoolTalk) often employ a user-viewable and user-updated address book to facilitate the process of locating and obtaining the correct Internet address; however, they require active user input. This is difficult in the case where users obtain access to the Internet via dial-up connections, and thus have changing Internet addresses. Still another problem with the prior art is that the applications described above provide only a limited degree of functionality; that is, they are

4

limited to either chat, video conferencing, or the like. As such, they are not capable of establishing a connection between any type of user system and remote device.

One prior art system is described in the copending previously filed patent application, assigned to the assignee of the present invention, entitled "A Method and System for Hosting an Internet Web Site on a Digital Camera," Eric C. Anderson and others, Ser. No. 09/044,644. This prior art system presents one solution to the problem of gaining remote access to those digital devices where the location and Internet address of the device are highly changeable. This prior art system incorporates a Web server into the digital device, specifically a digital camera. However, the disadvantage to this prior art system is that the Web server consumes valuable memory and computational resources in the digital camera. In addition, because of the limited memory in a device such as a digital camera, the Web server is not as powerful as a Web server on a server computer system.

Thus, a need exists for an inexpensive and powerful method for implementing remote access to digital devices, such as digital cameras, where the location and Internet address of the device are highly changeable. A further need exists for an intuitive, simple protocol for presenting the device's functionality and capabilities to users. In addition, a need exists for a method of efficiently administering a plurality of separate devices. A need also exists for an efficient process of obtaining the address of the device that is transparent to the viewer. The present invention provides a novel solution to the above needs.

SUMMARY OF THE INVENTION

The present invention provides an inexpensive and powerful method for implementing remote access to digital devices, such as digital cameras, where the location and Internet address of the device are highly changeable. The present invention further provides an intuitive, simple protocol for presenting the device's functionality and capabilities to users. In addition, the present invention provides a method of efficiently administering a plurality of separate devices. The present invention also provides an efficient process of obtaining the address of a device that is transparent to the user.

In one embodiment, the present invention is an executable program for accessing a digital camera via a communication network using a Web server on a server computer system and a Web browser (or a program of similar function) on a client computer system that are communicatively coupled via the Internet. The address of the digital camera is registered in an executable program on the server computer system. The executable program is accessed by the client computer system. The executable program connects the digital camera and the server computer system. The executable program enables the client computer system and the digital camera to communicate using any protocol used by these devices, thus allowing data (e.g., images) acquired by the digital camera to be transferred to the client computer system.

The executable program can be implemented in a variety of forms. For example, the executable program can be a Java script. Alternatively, the executable program can be a cgi-bin (Common Gateway Interface-binaries).

For example, in the case of a digital camera, the executable program directly communicates commands from the client computer system to the digital camera when both the client computer system and the digital camera are on-line at the same time. Alternatively, if the digital camera is not

5

on-line, the commands from the client computer system are first stored in the server computer system, and then later communicated by the executable program to the digital camera after a connection between the server computer system and the digital camera is made. The capability to store and then forward commands and data is not limited to a digital camera application nor is it limited to a particular data storage format. The data storage format can be any format that is understood by both the client computer system and the digital device.

Images and any other data acquired by the digital camera are accessed by the server computer system using the executable program and directly transferred to the client computer system when both the client computer system and the digital camera are on-line at the same time. Alternatively, if the client computer system is not on-line, the data are first stored by the server computer system, and then later communicated to the client computer system after a connection between the server computer system and the client computer system is made.

It should be noted, however, that the present invention can be readily modified to function in other embodiments, such as, for example, hand-held digital devices, lap top personal computers, and the like, which require an efficient process of obtaining the address of a device that is transparent to the user.

These and other objects and advantages of the present invention will become obvious to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments which are illustrated in the various drawing figures.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not by way of limitation, in the figures of the accompanying drawings in which like reference numerals refer to similar elements and in which:

FIG. 1A shows a block diagram of a remote access system via the Internet in accordance with one embodiment of the present invention.

FIG. 1B shows a block diagram of a remote access system via a Local Area Network in accordance with one embodiment of the present invention.

FIG. 1C shows a block diagram of a digital camera coupled to the Internet via an Internet Service Provider.

FIG. 2 shows a general purpose computer system upon which embodiments of the present invention may be practiced.

FIG. 3 shows a block diagram of a digital camera for use in accordance with the present invention.

FIG. 4 shows a block diagram of a computer system of a digital camera in accordance with one preferred embodiment of the present invention.

FIG. 5 shows a memory map of a dynamic random access memory of a digital camera in accordance with one embodiment of the present invention.

FIG. 6 shows a diagram of the connectivity and application software of a digital camera in accordance with one embodiment of the present invention.

FIG. 7 is a flowchart of a process for remotely accessing a digital camera implemented by an executable program in accordance with one embodiment of the present invention.

FIG. 8 is a flowchart of a process employed by a digital camera for remote access in accordance with one embodiment of the present invention.

6

FIG. 9 is a flowchart of a process employed by a client computer system for remote access of a digital camera in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In the following detailed description of the present invention, numerous specific details are set forth in order to enable one of ordinary skill in the art to make and use the invention, and are provided in the context of a patent application and its requirements. Although the present invention will be described in the context of a digital camera, various modifications to the preferred embodiment will be readily apparent to those skilled in the art, and the generic principles herein may be applied to other embodiments. That is, any digital device which displays data, images, icons and/or other items, could incorporate the features described below and that device would be within the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

Some portions of the detailed descriptions which follow are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computer memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is here, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computer system. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, bytes, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that throughout the present invention, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes (e.g., the processes of FIGS. 7, 8 and 9) of a computer system or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present invention provides a method for making a digital device (e.g., a digital camera) and its internally stored data remotely accessible over a communication network such as the Internet or a Local Area Network (LAN). The present invention is an executable program placed on a server computer system (specifically, a Web server) that implements commands initiated by a client (or user) using a client computer system with a Web browser or a program of similar function. The present invention enables the digital camera to be set to continuously take pictures of scenes/

items of interest and allow a client to access those pictures at any time. The present invention allows the client to access the digital camera from virtually an unlimited number of locations and with the camera in virtually any location.

Referring now to FIG. 1A, a block diagram of communication network 100 is shown. Communication network 100 provides a method in accordance with one embodiment of the present invention which implements remote access to camera 300 and its internally stored data. Communication network 100 includes camera 300, Internet Service Provider (ISP) 110, Internet Service Provider 115, client (or user) computer system 120, and server computer system 190. ISP 110 and ISP 115 are both directly coupled to the Internet 150. Client computer system 120 includes Web browser 121 or a program of similar function, and server computer system 190 includes Web server 161. Web browser 121 interprets HTML (HyperText Mark-up Language) documents and other data retrieved by Web server 161.

In the present embodiment of the present invention, an executable program resides on server computer system 190, specifically on Web server 161. The executable program implements and manages the connection between server computer system 190, client computer system 120, and camera 300. The executable program can be implemented as a Java servlet, as a cgi-bin (Common Gateway Interface-binaries), or as a similar type of application.

With reference still to FIG. 1A, client computer system 120 is communicatively coupled to ISP 115 via a POTS (plain old telephone system) dial-up connection. Client computer system 120 is coupled to the Internet 150 via one of a bank of modems maintained on the premises of ISP 115. ISP 115 is coupled directly to the Internet via an all-digital connection (e.g., a T1 line). However, other means of coupling client computer system 120 to the Internet 150 may be used in accordance with the present invention.

As depicted in FIG. 1A, camera 300 is communicatively coupled to server computer system 190 via the Internet 150 using a dial-up connection to ISP 110 via a POTS line. Digital camera 300 accesses ISP 110 using a modem, coupling to one of a bank of modems maintained on the premises of ISP 110. ISP 110 is in turn coupled directly to the Internet 150 via an all-digital connection. However, other means of coupling camera 300 to the Internet 150 may be used in accordance with the present invention.

With reference still to FIG. 1A, it should be further appreciated that while communication network 100 shows camera 300 coupling to Internet 150 via one ISP (e.g., ISP 110) and user 120 coupling to Internet 150 via a separate ISP (e.g., ISP 115), user 120 and camera 300 could be coupled to Internet 150 through a single ISP. In such a case, user 120 and camera 300 would be coupled to two separate access ports (e.g., two separate modems out of a bank of modems) of the same ISP.

With reference now to FIG. 1B, in another embodiment of the present invention, the communication network is comprised of Local Area Network (LAN) 170. For example, LAN 170 may be a communication network located within a firewall of an organization or corporation. Client (or user) computer system 171 and server computer system 172 are communicatively coupled via communication line 173a. Client computer system 171 includes an application that is analogous to a Web browser for interpreting HTML documents and other data. Similarly, server computer system 172 includes an application analogous to a Web server for retrieving HTML documents and other data. Camera 300 can be coupled to server computer system 172 through any

of a variety of means known in the art. For example, camera 300 can be coupled to server computer system 172 via communication line 173b of LAN 170. This coupling can be accomplished over any network protocol that supports a persistent network connection, such as TCP (Transmission Control Protocol), NetBIOS, IPX (Internet Packet Exchange), and LU6.2, and link layers protocols such as Ethernet, token ring, and ATM (Asynchronous Transfer Mode). Alternatively, camera 300 can be coupled to server computer system 172 via an input/output port (e.g., a serial port) of server computer system 172.

Referring now to FIG. 1C, a more detailed diagram of camera 300 coupled to the Internet 150 is shown. Camera 300 is coupled to an external modem 101. Camera 300 is coupled to modem 101 via any of several communications means (e.g., Universal Serial Bus, infrared link, and the like). Modem 101 is in turn coupled to a POTS telephone jack 102 at the camera's location. Telephone jack 102 couples modem 101 to one of modems 103 of ISP 110 via the telephone company's local loop. ISP 110 is directly coupled to the Internet 150 via an all digital connection (e.g., a T1 line).

Continuing with reference to FIG. 1C, modem 101 is shown as an external modem. However, the functionality of modem 101 can be implemented directly within the electronics of camera 300 (e.g., via a modem application-specific integrated circuit or ASIC), or alternatively can be implemented as a software-only modem executing on a computer within camera 300. As such, it should be appreciated that, at the hardware connectivity level, modem 101 can take several forms. For example, a wireless modem can be used in which case the camera is not connected via an external wire to any land line. Alternatively, there may be applications in which camera 300 includes suitable electronic components enabling a connection to a conventional computer system network (e.g., Ethernet, AppleTalk, and the like), which is in turn directly connected to the Internet (e.g., via a gateway, a firewall, and the like), thereby doing away with the requirement for an ISP. Hence, it should be appreciated that the present invention is not limited to any particular method of accessing the Internet 150.

Refer now to FIG. 2 which illustrates server computer system 190 upon which embodiments of the present invention may be practiced (the following discussion is also pertinent to a client computer system). In general, server computer system 190 comprises bus 200 for communicating information, processor 201 coupled with bus 200 for processing information and instructions, random access memory 202 coupled with bus 200 for storing information and instructions for processor 201, read-only memory 203 coupled with bus 200 for storing static information and instructions for processor 201, data storage device 204 such as a magnetic or optical disk and disk drive coupled with bus 200 for storing information and instructions, optional display device 205 coupled to bus 200 for displaying information to the computer user, optional alphanumeric input device 206 including alphanumeric and function keys coupled to bus 200 for communicating information and command selections to processor 201, optional cursor control device 207 coupled to bus 200 for communicating user input information and command selections to processor 201, and network interface card (NIC) 208 coupled to bus 200 for communicating from a communication network to processor 201.

Display device 205 of FIG. 2 utilized with server computer system 190 of the present invention may be a liquid crystal device, cathode ray tube, or other display device

suitable for creating graphic images and alphanumeric characters recognizable to the user. Cursor control device 207 allows the computer user to dynamically signal the two-dimensional movement of a visible symbol (pointer) on a display screen of display device 205. Many implementations of the cursor control device are known in the art including a trackball, mouse, joystick or special keys on alphanumeric input device 206 capable of signaling movement of a given direction or manner of displacement. It is to be appreciated that the cursor control 207 also may be directed and/or activated via input from the keyboard using special keys and key sequence commands. Alternatively, the cursor may be directed and/or activated via input from a number of specially adapted cursor directing devices.

Referring now to FIG. 3, a block diagram of digital camera 300 is shown for use in accordance with the present invention. Camera 300 preferably comprises imaging device 314, system bus 316 and computer 318. Imaging device 314 is optically coupled to object 312 and electrically coupled via system bus 316 to computer 318. Once a photographer has focused imaging device 314 on object 312 and, using a capture button or some other means, instructed camera 300 to capture an image of object 312, computer 318 commands imaging device 314 via system bus 316 to capture raw data representing object 312. The captured raw data are transferred over system bus 316 to computer 318, which performs various processing functions on the data before storing it in memory. System bus 316 also passes various status and control signals between imaging device 314 and computer 318.

Referring now to FIG. 4, a block diagram of one embodiment of computer 318 is shown. System bus 316 provides connection paths between imaging device 314, an optional power manager 442, central processing unit (CPU) 444, dynamic random-access memory (DRAM) 446, input/output interface (I/O) 448, non-volatile memory 450, and buffers/connectors 452. Removable memory 454 connects to system bus 316 via buffers/connectors 452. Alternatively, camera 300 may be implemented without removable memory 454 or buffers/connectors 452.

Power manager 442 communicates via line 466 with power supply 456 and coordinates power management operations for camera 300. CPU 444 typically includes a conventional processor device for controlling the operation of camera 300. In the present embodiment, CPU 444 is capable of concurrently running multiple software routines to control the various processes of camera 300 within a multi-threaded environment. DRAM 446 is a contiguous block of dynamic memory which may be selectively allocated to various storage functions. LCD controller 490 accesses DRAM 446 and transfers processed image data to LCD screen 402 for display.

I/O 448 is an interface device allowing communications to and from computer 318. I/O 448 permits an external device (not shown) to connect to and communicate with computer 318. I/O 448 also interfaces with a plurality of buttons and/or dials 404, and optional status LCD 406, which in addition to LCD screen 402, are the hardware elements of the camera's user interface 408.

Non-volatile memory 450, which may typically comprise a conventional read-only memory or flash memory, stores a set of computer-readable program instructions to control the operation of camera 300.

Referring now to FIG. 5, a memory map showing one embodiment of dynamic random access memory (DRAM) 446 is shown. In the present embodiment, DRAM 446 includes RAM disk 532, system area 534, and working memory 530.

RAM disk 532 is a memory area used for storing raw and compressed data and typically is organized in a "sectored" format similar to that of conventional hard disk drives. In the present embodiment, RAM disk 532 uses a well-known and standardized file system to permit external devices, via I/O 448 of FIG. 4, to readily recognize and access the data stored on RAM disk 532. System area 534 typically stores data regarding system errors (for example, why a system shutdown occurred) for use by CPU 444 (FIG. 4) upon a restart of computer 318 (FIG. 3).

Working memory 530 includes various stacks, data structures and variables used by CPU 444 while executing the software routines used within computer 318. Working memory 530 also includes several input buffers 538 for temporarily storing sets of raw data received from imaging device 314 (FIG. 3), and frame buffer 536 for storing data for display on LCD screen 402 (FIG. 4). In the present embodiment, each input buffer 538 and frame buffer 536 are split into two separate buffers (shown by the dashed lines) to improve the display speed of the digital camera and to prevent the tearing of the image in LCD screen 402.

Referring now to FIG. 6, a diagram of the connectivity and application software of camera 300 of FIG. 3 is shown. At the software level, computer 318 (FIG. 3) of camera 300 hosts any network protocol that supports a persistent network connection. This coupling can be accomplished over any network protocol that supports a persistent network connection, such as TCP/IP (Transmission Control Protocol/Internet Protocol) including Point-to-Point Protocol, NetBIOS, IPX, and LU6.2, and link layers protocols such as Ethernet, token ring, and ATM. Protocol stack 601 interfaces with the communications hardware 602 (e.g., a modem) of camera 300 and the application layer 603. The bottom of protocol stack 601 includes communication hardware interface drivers which interface directly with the various communications hardware with which camera 300 must function (e.g., a Universal Serial Bus and the like). Applications layer 603, protocol stack 601, and operating system 604 are installed as software modules in DRAM 446 (FIG. 4) of camera 300. Software applications within applications layer 603 interface with operating system 604. Operating system 604 controls the hardware functionality of camera 300 (e.g., taking pictures, storing pictures, and the like) via camera hardware 605. Incoming data 620, such as HTTP (Hypertext Transfer Protocol) requests and the like, are received and outgoing data 630, such as HTML (Hypertext Markup Language) files and the like, are transferred to and from camera 300 via protocol stack 601 and communications hardware 602. Web browser 121 of FIG. 1A (or a program of similar function) can process data files, launch plug-ins, and run Java applets that communicate with camera 300 in a variety of methods in addition to those methods involving an exchange of HTML files.

FIG. 7 illustrates an executable program 700 for client computer system 120 (specifically, Web browser 121 of FIG. 1A or a program of similar function) to remotely access camera 300 (FIG. 3), where executable program 700 is implemented in accordance with the present invention as program instructions stored in computer-readable memory units (e.g., read-only memory 203) and implemented by processor 201 of server computer system 190 of FIG. 1A (specifically, by Web server 161). Executable program 700 performs functions both for and in response to Web browser 121 (or a program of similar function) and camera 300. The description below first discusses the steps associated with setting up a connection between executable program 700 and camera 300, then discusses the steps associated with

11

setting up a connection between executable program 700 and Web browser 121 (or a program of similar function), however, the present invention is not limited by the order in which these steps are presented.

In the present embodiment, executable program 700 is identified and accessed by its own unique address, commonly referred to as an URL (Unified Resource Locator), as is well known in the art. The URL for executable program 700 fully describes where it resides on a communication network (e.g., the Internet 150) and how it is accessed. In the present embodiment, included in the URL for executable program 700 is the name of camera 300. Accordingly, in one embodiment using a servlet for executable program 700, a standard format for a URL is: `http://webserverHostName/cameraServletWellKnownName/cameraName`.

In step 705, executable program 700 receives and accepts a connection request from camera 300. Executable program 700 runs constantly on Web server 161 and is configured to listen for connection requests on a plurality of communication protocols (e.g., TCP, NetBIOS, and the like). In the present embodiment, camera 300 is connected to executable program 700 via Web server 161 as described in conjunction with FIG. 1A, and opens the connection using the connectivity and application software described in conjunction with FIG. 6.

In step 710, executable program 700 receives and reads registration information from camera 300. As mentioned above, executable program 700 is configured to communicate using a number of different communication protocols. Such registration information includes the name of the camera and authentication information such as security information and account information. Executable program 700 uses the camera name to identify the camera and locate it in response to a client request.

In step 720, executable program 700 compares the registration information with a predefined access control list to determine if camera 300 is a camera for which Web server 161 is to provide support and service. If not, executable program 700 closes the connection between Web server 161 and camera 300.

In step 725, upon successful completion of step 720, executable program 700 registers camera 300 and stores the camera's name and associated requirements, such as security and account information. Executable program 700 also sends a message that camera 300 is registered. At this point, the connection between executable program 700 can be either terminated or maintained at the option of the camera's operator. If the connection is terminated, the registration information is maintained by Web server 161 and can be later accessed by executable program 700 when a subsequent connection is made with camera 300.

In step 730, executable program 700 accepts the connection request from camera 300 and thus has a persistent and long term connection with camera 300. As described above, the connection can be an ongoing connection maintained from the time when camera 300 was first registered. New connections can be made in the future whenever camera 300 reinitiates the registration protocol. Once camera 300 and executable program 700 have established a connection, they then wait until a client also makes a connection to access the camera. However, as will be described below in conjunction with step 770, it is not necessary for there to be an open connection between the Web server and the camera at the same time that there is an open connection between the Web browser (or a program of similar function) and Web server to accomplish remote access of the camera in accordance with the present invention.

12

In step 740, executable program 700 receives and accepts a request for a connection from a client. The client enters the URL of executable program 700, including the name of the camera (e.g., camera 300) to which access is desired, into a Web browser (e.g., Web browser 121 on client computer system 120 of FIG. 1A or a program of similar function). Using standard communication protocols such as TCP/IP, Web server 161 is queried with the URL for executable program 700. Web server 161 recognizes the URL and makes the connection to executable program 700. Thus, the present invention establishes a single location identified by a known URL where the client always goes to connect to the camera, no matter where the camera is or where the client is. (For those cases in which TCP/IP is not available—for example, when the device is not attached to the Internet or the like—camera 300 connects to executable program 700 directly using a well-known address that is associated with the protocol being used, such as a NetBIOS name.)

In step 745, executable program 700 then assumes control over the TCP/IP connection between Web browser 121 (or a program of similar function) and Web server 161. Executable program 700 establishes a persistent and long term connection between the browser and server. That is, the connection between Web browser 121 (or a program of similar function) and Web server 161 is kept open by executable program 700. As will be described below in conjunction with step 770, it is not necessary for there to be an open connection between the client computer system and the Web server at the same time that there is an open connection between the Web server and the camera to accomplish remote access of the camera in accordance with the present invention.

In step 750, executable program 700 next determines if camera 300 is registered as discussed in conjunction with step 735. If camera 300 is not registered, executable program 700 sends an appropriate message to the client to indicate such.

In step 760, if the camera is registered, executable program 700 validates the required access information provided by the client against the security and account information provided when camera 300 was registered. For example, executable program 700 validates whether the client is utilizing an authorized password or user name. If not, executable program 700 transmits an appropriate message to the client. The present invention can optionally provide additional services related to security or account information. For example, it could control the type of access a client is permitted based on the authentication information received from the client, or it could verify credit information and bill the client for services requiring payment.

In step 765, upon satisfactory completion of step 760, executable program 700 extends the connection from Web browser 121 (or a program of similar function) to camera 300 if there is an established connection to the camera as described in conjunction with step 730. Hence, a client using Web browser 121 or a program of similar function has direct, remote access to camera 300 via executable program 700 in Web server 161.

In step 770, executable program 700 forwards commands from a client to camera 300, and forwards images and other data from camera 300 to the client via Web server 161 and Web browser 121 (or a program of similar function). That is, executable program 700 enables a direct communication between the client computer system and the camera allowing the client to remotely access and manage the camera. If the client and the camera are both concurrently connected to

13

executable program 700, then the client immediately receives the data, and camera 300 immediately executes any commands from the client.

However, if Web browser 121 (or a program of similar function) and camera 300 are not each connected at the same time, remote access to the camera is still accomplished in accordance with the present invention. If camera 300 is not on-line, a client uses Web browser 121 (or a program of similar function) to access Web server 161 and executable program 700. The client transmits commands, and executable program 700 stores the commands on Web server 161. The client may then close the connection to the Web server. Subsequently, when camera 300 opens a connection to Web server 161 and executable program 700, executable program 700 retrieves the commands and forwards them to the camera. Camera 300 downloads the commands and executes them. The results from executing the commands are then sent to executable program 700, which stores them on Web server 161 until they are retrieved by the client.

Similarly, if camera 300 establishes a connection with Web server 161 and executable program 700 but the client is not on-line, the camera can, for example, download images and any other data that executable program 700 stores on Web server 161. Camera 300 may then close the connection to the Web server. The client then later makes a connection to Web server 161 and executable program 700, which retrieves the data and forwards it to the client. The client can also enter commands at this time, which are stored by executable program 700 as described above.

In summary, with references to FIG. 1A and FIG. 7, the client, via Web browser 121 or a program of similar function, Web server 161 and executable program 700, accesses camera 300 to request and retrieve data. Web browser 121 or a program of similar function inserts the Internet address inside the data request (e.g., a HTTP request) and sends the request to Web server 161. Web server 161 receives the data request and associates the request with executable program 700, which in turn assumes the connection between Web server 161 and Web browser 121 (or a program of similar function), and which also establishes a connection between Web browser 121 (or a program of similar function) and camera 300. Executable program 700 subsequently forwards commands from the client to camera 300, and retrieves the requested data (e.g., a HTML file) containing the data and sends it back to Web browser 121 (or a program of similar function). Web browser 121 (or a program of similar function) then interprets the commands and displays the resulting image. The process of accessing a data file from a Web server is commonly referred to as accessing a Web page. Similarly, the process of sending data files from a Web server to a Web browser is commonly referred to as sending a Web page.

Thus, as described above, the present invention provides an intuitive and easy-to-use interface enabling remote access between a client and a camera. By functioning with widely used and familiar Web browsers (or programs of similar function) using standard format URLs to identify the executable program and camera, the present invention provides a simple and familiar interface for accessing the camera. By registering the camera and using an unchanging URL, name to identify the executable program and the camera, the present invention enables the client to locate and access the camera from any remote location no matter where the camera is located. In addition, by using a Java servlet or a cgi-bin for the executable program, the present invention is supported by commonly used Web servers and is readily implemented.

14

Executable program 700 is located on the Web server and not on camera 300, so it does not require additional and substantial memory dedicated to enabling remote access. As such, the present invention permits remote access within the constraints of the size of the camera. In addition, in accordance with the present invention, camera 300 does not require a separate, external computer system (e.g., a personal computer system) for connecting to ISP 110 (FIG. 1A) or for implementing commands and transmitting data, thus providing an inexpensive method for providing remote access to cameras.

Also, by locating the present invention on a Web server (e.g., Web server 161 of FIG. 1A), the Web server becomes a focal point for accessing and managing a plurality of cameras that otherwise would have to be managed and configured separately. For example, executable program 700 on Web server 161 could be updated with new software, and in effect all cameras accessed through that executable program would automatically be updated as well. As another example, executable program 700 could be configured to compile data regarding interactions between clients and all cameras accessed by the executable program. In another example, in accordance with the present invention, a client needs to go only to a single location to determine which of a plurality of cameras served by the executable program have data that have been downloaded to the Web server. Thus, instead of having to access a number of cameras separately, the present invention establishes a single location from which a client can access information about several cameras.

By functioning with a Web-based interface and widely used Web browsers (or programs of similar function), the present invention provides a simple, intuitive, and familiar interface for accessing camera 300's functionality. Accordingly, camera 300's controls and functions are intuitively easy to utilize. Since Web pages and their associated controls (e.g., push buttons, data entry fields, and the like) are very familiar to most users, the remote access functionality of camera 300 can be utilized without requiring a extensive learning period for new users. For example, a consumer purchasing a remotely accessible camera is typically able to easily and immediately use the remote accessibility functions with minimal set-up.

FIG. 8 illustrates a process 800 for remotely accessing camera 300 (FIG. 3), where process 800 is implemented as program instructions stored in computer-readable memory units (e.g., non-volatile memory 450 of FIG. 4) and implemented by CPU 444 (FIG. 4) of camera 300 in accordance with the present invention. FIG. 8 illustrates the process for remotely accessing a camera from the perspective of the actions performed by the camera and the camera operator in accordance with the present invention.

In step 805, camera 300 of FIG. 3 opens a connection to communication network 100 of FIG. 1A. In the present embodiment, camera 300 is coupled to server computer system 190 (specifically, Web server 161) as described in conjunction with FIG. 1A, and opens the connection using the connectivity and application software described in conjunction with FIG. 6. As described in conjunction with FIG. 1C, in the present embodiment, camera 300 couples directly to the telephone system such that a separate and dedicated computer system (e.g., a personal computer system) is not necessary.

In step 810, with the connection made to Web server 161, camera 300 requests a connection to executable program 700 (FIG. 7). For cases in which camera 300 is accessing

15

executable program 700 via, for example, a TCP/IP network such as the Internet 150, then executable program 700 is identified with a URL that is used by the camera to access the executable program. For those cases in which TCP/IP is not available (e.g., when the device is not attached to the Internet or the like), camera 300 connects to executable program 700 directly using a well-known address that is associated with the protocol being used, such as a NetBIOS name.

In step 820, with the camera connected to the executable program, camera 300 registers with executable program 700. For example, camera 300 provides information including an identification name and authentication information such as a password and account information. The information is electronically transmitted from camera 300 and read by executable program 700. Based on this information, the connection between camera 300 and executable program 700 is established if the camera is of the type that is designated to be supported by Web server 161.

In step 830, camera 300 and executable program 700 are linked via a persistent and long term connection; that is, the connection remains open awaiting a client to request access to the camera via Web server 161. As discussed above in conjunction with FIG. 7, it is not necessary for the camera and a client to be connected at the same time to executable program 700.

In step 835, camera 300 receives commands from and transmits data to a client using a Web browser or a program of similar function on a client computer system (e.g., client computer system 120 and Web browser 121 of FIG. 1A or a program of similar function). As described above in conjunction with FIG. 7, the commands and data can be transmitted through an active connection or stored on the Web server.

In the present embodiment, camera 300 is provided with several different operating modes for supporting various camera functions. In capture mode, camera 300 supports the actions of preparing to capture an image and of capturing an image. In review mode, camera 300 supports the actions of reviewing camera contents, editing and sorting images, and printing and transferring images. In play mode, camera 300 allows the client to view screen-sized images in the orientation that the image was captured. Play mode also allows the client to hear recorded sound associated to a displayed image, and to play back sequential groupings of images, which may comprise time lapse, slide show, and burst image images. The client preferably switches between the capture, review, and play modes.

Camera 300 is capable of implementing a wide variety of remote access and remote imaging/surveillance applications. In the present embodiment, camera 300 only records successive images for remote access by the client. The images are loaded into the camera's memory on a first-in, first-out (FIFO) basis, with the earliest recorded image being replaced by the latest recorded image. The number of images available to the client depends upon the amount of installed memory in the camera.

With reference still to step 835 of FIG. 8, when the client and the camera are connected to the Web server at the same time, the commands can be transmitted directly from Web browser 121 (or a program of similar function) to camera 300 via executable program 700 on Web server 161, and camera 300 then executes the commands on-line. Alternatively, when camera 300 is not connected to executable program 700, a client can store commands on Web server 161 by accessing the Web server in a normal fashion,

16

then entering the commands to be stored via executable program 700. Camera 300 then executes the commands when it subsequently connects with executable program 700.

Also in step 835, camera 300 transmits data to a client. Similar to the above, data from camera 300 can be transmitted directly to Web browser 121 or a program of similar function via executable program 700 on Web server 161, when the client and the camera are connected to the Web server at the same time. Alternatively, when a client is not connected to executable program 700, the data are stored on Web server 161 by executable program 700, which then retrieves and transmits the data to a client when the client subsequently connects with executable program 700.

This process of accessing camera 300 from Web browser 121 or a program of similar function occurs transparently with respect to the client. In a typical case, for example, the client types the URL for executable program 700 (which includes the name of camera 300) into Web browser 121 (or a program of similar function) and his "enter" or "return". In accordance with the present invention, the next Web page the client views is the image generated by the data returned from camera 300. Beyond entering the URL for executable program 700 including camera 300, no further action from the client is required in order to access the Web pages hosted by camera 300.

FIG. 9 illustrates a process 900 for remotely accessing camera 300 (FIG. 3), where process 900 is implemented as program instructions stored in computer-readable memory units (e.g., read-only memory) and implemented by the central processor of client computer system 120 (specifically, Web browser 121 or a program of similar function) of FIG. 1A. FIG. 9 illustrates the process for remotely accessing a camera from the perspective of the actions performed by the client in accordance with the present invention.

In step 905, the client opens a network connection such as a dial-up connection to ISP 115 of FIG. 1A.

In step 910, the client enters into a Web browser (e.g., Web browser 121 on client computer system 120 of FIG. 1A or a program of similar function) the URL of executable program 700, including the name of the camera (e.g., camera 300) to which access is desired. Alternatively, the client enters the URL of executable program 700 only. (For those cases in which TCP/IP is not available—for example, when the device is not attached to the Internet or the like—camera 300 connects to executable program 700 directly using a well-known address that is associated with the protocol being used, such as a NetBIOS name.) Using standard communication protocols such as TCP/IP, Web server 161 is queried with the URL for executable program 700. Web server 161 recognizes the URL and makes the connection to executable program 700. A persistent and long term two-way connection is opened between the client on Web browser 121 (or a program of similar function) and executable program 700.

In step 915, the client enters the authentication information required to gain access to executable program 700 and camera 300.

In step 920, as explained above, from his/her remote location the client sends commands to camera 300 and receives images from the camera. As explained above, the client and the camera do not have to be connected to the Web server at the same time to enable remote access. Depending on the particular application, the Web page for camera 300 can include control buttons, data entry fields, drop-down menus, and the like.

17

Thus, executable program 700 enables the client to access from a remote location the functional controls of camera 300 as well as the images and other data acquired by the camera.

Pseudo-Code Sections A, B, C, D and E below provide additional details regarding processes 700, 800 and 900 of FIGS. 7, 8 and 9. Pseudo-Codes Sections A through E represent the method of one embodiment of the present invention. However, it is appreciated that other embodiments are possible in accordance with the present invention.

18

With references to FIGS. 7 and 9, the pseudo-code for the connection of a client to the server computer system, specifically to executable program 700, is described in accordance with one embodiment of the present invention in Pseudo-Code Section A below.

Pseudo-Code Section A: Client Connection Setup

```

1. Client sends HTTP Post command to http://serverName/gateway device="deviceName"
   Note: authentication/security data is optional and is handled by the browser.
2. Client receives HTTP response from Gateway
3. If response is OK or OKDataPresent
{
    Client has TCP connection it can use to send/receive anything to/from the device
    The protocol used can be any that the Client & Device agree upon.
}
else
{
    The response is failed or DataPresent.
    The TCP connection is closed:
}

```

Note: the OKDataPresent and DataPresent responses are informational. The Client at any time on separate connections may issue a CacheData or GetCache command to the Gateway. See algorithms C.

30 With references to FIGS. 7 and 8, the pseudo-code for the connection of a device (e.g., digital camera 300 of FIG. 1A) to the server computer system, specifically to executable program 700, is described in accordance with one embodiment of the present invention in Pseudo-Code Section B below.

Pseudo-Code Section B: Device Connection Setup/Registration

```

1. Device establishes a connection to the Gateway using any networking protocol
   supported by the Gateway (TCP, NetBIOS, IPX, 802.2, etc) using a known address.
2. Device sends a Register request to the Gateway on the connection passing
   its name (and optionally authentication/security information).
3. Device receives response from Gateway
4. If the response is OK or OKDataPresent
{
    The Device may use the existing connection to wait for requests from clients
    or the Device may optionally close the connection.
}
else
{
    The response is Failed or DataPresent
    The connection is closed.
}

```

Note: the OKDataPresent and DataPresent responses are informational. The Device at any time on separate connections may issue a CacheData or GetCache command to the Gateway. See algorithm D.

60

With reference to FIG. 7, the pseudo-code for executable program 700 in response to a client connection request is described in accordance with one embodiment of the present invention in Pseudo-Code Section C below.

65

Pseudo-Code Section C: Gateway Handles Client Connection Request

```

Start:
    Wait for incoming requests from clients
    if its an HTTP Post command with parameters CacheData and deviceName
    {
        go to CacheData
    }
    else if its an HTTP Get command with parameters GetCache and deviceName
    {
        go to GetCache
    }
    if its an HTTP Post command from a client with the name of device
    {
        go to Connect
    }
    else
    {
        go to Fail
    }
Connect:
    Look up device by name in registry
    if the device entry is found
    {
        if there is a device connection id in the entry
        {
            if the entry is not busy
                go to Success
            else go to Fail
        }
        else
        {
            Attempt to establish connection with device at last known address
            if connection established
            {
                go to Success
            }
            else go to Fail
        }
    }
    else
    {
        send Fail response
        close connection
        go to Start
    }
Fail:
    if there is data in the cache
    {
        set response to DataPresent
    }
    else
    {
        set response to Fail
    }
    send response
    close connection
    go to Start
Success:
    if there is data in the cache
    {
        set response to OkDataPresent
    }
    else
    {
        set response to Ok
    }
    store connection id of the client in the registry entry
    go to Start
CacheData:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        store data in Cache for device with identification information

```

-continued

```

    send Ok response
    close connection
    go to Start;
}
GetCache:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        send data in Cache for this client
        close connection
        go to Start
    }
}

```

With reference to FIG. 7, the pseudo-code for executable program 700 in response to a device connection request is described in accordance with one embodiment of the present invention in Pseudo-Code Section D below.

Pseudo-Code Section D: Gateway Handles Device Requests

```

Start:
    Wait for incoming requests from devices
    Receive incoming request
    if its a Register request
    {
        go to Register
    }
    else if its a CacheData request
    {
        go to CacheData
    }
    else if its a GetCache request:
    {
        go to GetCache
    }
    else
    {
        return Failed
        close connection
        go to Start
    }
Register:
    Note: The gateway may optionally authenticate device
    Get device name from Register request
    look up name in table
    if name is found
    {
        go to Success
    }
    else
    {
        The gateway may optionally add the name or reject the
        registration attempt, then go to Success or Fail
        respectively
    }
Success:
    check for the presence of cached client data
    if cached data is present
    {
        set response to OkDataPresent;
    }
    else
    {
        set response to Ok
    }
}

```

-continued

```

    Store a connection id for the incoming connection in
    the registry entry for the device.
    send response
    go to Start
Fail:
    if cached data from clients is present for this device
    {
        set response to DataPresent
    }
    else
    {
        set response to Fail
    }
    send response
    close connection
    go to Start
CacheData:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        store data in Cache for client
        send Ok response
        close connection
        go to Start;
    }
GetCache:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        send data in Cache for device to the device
        close connection
        go to Start
    }
}

```

With reference to FIG. 7, the pseudo-code for executable program 700, specifically the handling of data when an open connection is present between the device and the client

computer system, is described in accordance with one embodiment of the present invention in Pseudo-Code Section E below.

Pseudo-Code Section E: Gateway Handles Data on Existing Connections

```

Start:
  Wait for data
  Map the incoming connection id to its partner connection id
  if there is incoming data
  {
    Send the data out on the partner connection id
    go to Start
  }
  else if the connection closed
  {
    if it is a client connection
    {
      Remove the client's connection id from the
      device's entry in the registry.
    }
    else
    {
      Remove both the client and device connection ids
      from the device's entry in the registry.
      Close the client's connection
      go to Start
    }
  }
  go to Start
  
```

As described above, the remote accessibility of camera **300** provides for many new applications of digital imagery. One such application involves setting up camera **300** at some remote location and using it to take pictures at successive intervals. These pictures would be accessed via the Internet **150** as they are taken. The interval can be adjusted (e.g., more or less pictures per minute) in response to commands entered by a client via a Web browser (e.g., Web browser **121** of FIG. 1A or a program of similar function).

Another application involves using camera **300** in conjunction with a motion detector. When used in conjunction with a motion detector, camera **300** can be configured to capture an image in response to receiving a signal from the motion detector (e.g., detecting the motion of an intruder), thereby taking a picture of whatever triggered the detector's signal output. Alternatively, camera **300** can detect motion by simply comparing successive images to detect changes between them, thereby dispensing with the need for a separate motion detector.

Yet another application involves using camera **300** in conjunction with a remote aiming device. Camera **300** can be mounted on a remotely operated aiming device (e.g., a motorized tripod). The aiming device is controlled via the Internet **150** in the same manner the camera is controlled via the Internet **150**. Alternatively, camera **300** could be coupled to control the remote aiming device directly. The remote aiming device allows a client to control the field of view of the camera **300** in the same manner the client controls other functionality (e.g., picture resolution, picture interval, and the like).

In this manner, executable program **700** of the present invention is able to implement sophisticated remote surveillance of the type previously performed by expensive, prior art closed circuit television devices. Unlike the prior art, however, executable program **700** is inexpensive and relatively simple to implement.

Thus, the present invention provides a method for making a digital camera and its internally stored data remotely

accessible. The present invention enables the digital camera to be set to continuously take pictures of scenes and items of interest and to allow a user to access those pictures at any time. The present invention implements remote accessibility via a communication network such as the Internet, thus allowing the user to access the digital camera from virtually an unlimited number of locations and with the camera in virtually any location.

A digital camera in accordance with the present invention does not require a separate, external computer system (e.g., a personal computer system) for Internet connectivity, thus providing an inexpensive method for making remotely accessible digital cameras widely available. In addition, a digital camera in accordance with the present invention is accessed via the widely used and very familiar Web browser (or a program of similar function). By functioning with typical, widely used Web browsers (or programs of similar function), the present invention provides a simple, intuitive, and familiar interface for remotely accessing the digital camera's functionality and capabilities. In so doing, the controls and functions of the digital camera are intuitively easy to utilize, and do not require an extensive learning period for new users. The present invention also provides an efficient and user-transparent process of obtaining the address of a digital camera. Also, the present invention provides a method for efficiently administering a plurality of separate digital cameras.

Although the present invention is described in the context of a digital camera, it is not limited to this embodiment. Hence, the present invention does not provide only a limited degree of functionality as in the prior art applications; that is, it is not limited to either chat or video conferencing, or the like. As such, the present invention is capable of establishing a connection between any type of client system and remote device.

The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.

What is claimed is:

1. A method for allowing a client computer to remotely access a digital image capture unit via a communication network wherein a server computer system and said client computer system are communicatively coupled with communication equipment, said method comprising the steps of:
 - a) allowing said digital image capture unit to establish communication with said server computer system over said network, wherein said digital image capture unit includes connectivity software that enables said digital image capture unit to establish a network connection with said server computer system;
 - b) receiving an address of said digital image capture unit and registering said address in an executable program on said server computer system, wherein said digital image capture unit automatically registers said address with said server computer system;
 - c) allowing said client computer system to access said executable program on said server computer system; and

25

- d) establishing direct communication between said client computer system and said digital image capture unit by communicating commands between said client computer system and said digital image capture unit via said executable program, such that data captured by said digital image capture unit is transferred to said client computer system via said server computer system.
2. The method of claim 1 wherein said communication network is the Internet.
3. The method of claim 1 wherein said communication network is a Local Area Network.
4. The method of claim 1 wherein said image capture unit is a digital camera.
5. The method of claim 1 wherein step a) further comprises communicating authentication information between said digital image capture unit and said executable program.
6. The method of claim 1 wherein said executable program is a Java servlet.
7. The method of claim 1 wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).
8. The method of claim 1 wherein step c) further comprises connecting said digital image capture unit to said server computer system via a Local Area Network.
9. The method of claim 1 wherein step c) further comprises connecting said digital image capture unit to said server computer system via an input/output port of said server computer system.
10. The method of claim 1 wherein step c) further comprises connecting said digital image capture unit to said server computer system via the Internet.
11. The method of claim 1 wherein step d) further comprises storing said commands in a memory unit of said server computer system and communicating said commands to said digital image capture unit at a time when a connection is made between said server computer system and said digital image capture unit.
12. The method of claim 1 further comprising the steps of:
- e) accessing via said executable program data acquired by said digital image capture unit; and
 - f) transferring said data from said digital image capture unit to said client computer system via said server computer system.
13. The method of claim 12 further comprising storing said data in a memory unit of said server computer system and communicating said data to said client computer system at a time when a connection is made between said server computer system and said client computer system.
14. A computer system comprising:
- a) a processor coupled to a bus; and
 - a memory unit coupled to said bus and having stored therein an executable program that when executed by said processor implements a method for allowing a client computer to remotely access a digital image capture unit via a communication network, said method comprising the steps of:
 - a) allowing said digital image capture unit to establish communication with said server computer system over said network, wherein said digital image capture unit includes connectivity software that enables said digital image capture unit to establish a network connection with said server computer system;
 - b) receiving an address of said digital image capture unit and registering said address with said executable program, wherein said digital image capture unit automatically registers said address with said server computer system;

26

- c) allowing said client computer system to access said executable program over said communication network; and
 - d) establishing direct communication between said client computer system and said digital image capture unit by communicating commands between said client computer system and said digital capture unit via said executable program.
15. The computer system of claim 14 wherein said computer system is a server computer system.
16. The computer system of claim 14 wherein said digital image capture unit is a digital camera.
17. The computer system of claim 14 wherein said executable program is a Java servlet.
18. The computer system of claim 14 wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).
19. The computer system of claim 14 wherein said digital image capture unit and said computer system are connected via a Local Area Network.
20. The computer system of claim 14 wherein said digital image capture unit and said computer system are connected via an input/output port of said computer system.
21. The computer system of claim 14 wherein said digital image capture unit and said computer system are connected via the Internet.
22. In a communication network comprising a server computer system and a client computer system communicatively coupled with communication equipment, a method for allowing said client computer to remotely access a digital camera via said communication network, said method comprising the steps of:
- a) allowing said digital camera to establish communication with said server computer system over said communication network, wherein said digital camera includes connectivity software that enables said digital camera to establish a network connection with said server computer system;
 - b) receiving an address of said digital camera and registering said address in an executable program on said server computer system, wherein said digital camera automatically registers said address with said server computer system;
 - c) allowing said client computer system to access said executable program on said server computer system;
 - d) establishing direct communication between said client computer system and said digital camera by communicating commands between said client computer system and said digital camera via said executable program;
 - e) accessing via said server computer system data acquired by said digital camera; and
 - f) transferring said data from said digital camera to said client computer system via said server computer system.
23. The method of claim 22 wherein said executable program is a Java servlet.
24. The method of claim 22 wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).
25. A method for allowing a client computer to remotely access a digital image capture device via a communication network wherein a server computer system and said client computer system are communicatively coupled with communication equipment, said method comprising the steps of:
- a) allowing said digital image capture device to establish communication with said server computer system over

27

said communication network, wherein said digital image capture device includes connectivity software that enables said digital image capture device to establish a network connection with said server computer system;

- b) receiving an address of said digital image capture device and registering said address in an executable program on said server computer system, wherein said digital image capture device automatically registers said address with said server computer system;
- c) allowing said client computer system to access said executable program on said server computer system; and

28

- d) establishing direct communication between said client computer system and said digital image capture device by communicating commands between said client computer system and said digital image capture device via said executable program, such that data from said digital image capture device is transferred to said client computer system via said server computer system.

26. The method of claim **25** wherein said executable program is a Java servlet.

27. The method of claim **25** wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).

* * * * *



US00662218B2

(12) **United States Patent**
Mighdoll et al.

(10) Patent No.: **US 6,662,218 B2**
(45) Date of Patent: ***Dec. 9, 2003**

(54) **METHOD OF TRANSCODING DOCUMENTS
IN A NETWORK ENVIRONMENT USING A
PROXY SERVER**

(75) Inventors: **Lee S. Mighdoll**, San Francisco, CA
(US); **Bruce A. Leak**, Palo Alto, CA
(US); **Stephen C. Perlman**, Mountain
View, CA (US); **Phillip Y. Goldman**,
Los Altos, CA (US)

(73) Assignee: **WebTV Networks, Inc.**, Mountain
View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 0 days.

This patent is subject to a terminal dis-
claimer.

(21) Appl. No.: **10/247,885**

(22) Filed: **Sep. 20, 2002**

(65) **Priority Publication Data**

US 2003/0014499 A1 Jan. 16, 2003

Related U.S. Application Data

(63) Continuation of application No. 09/343,067, filed on Jun.
29, 1999, now abandoned, which is a continuation of application
No. 08/656,924, filed on Jun. 3, 1996, now Pat. No.
5,918,013.

(51) **Int. Cl.** **G06F 15/16**

(52) **U.S. Cl.** **709/219; 709/203; 709/217;
709/228; 709/229; 709/246**

(58) **Field of Search** **709/200-203,
709/217-219, 227-229, 231-232, 246-247**

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,575,579 A 3/1986 Simon et al. 178/4
4,852,151 A 7/1989 Dittakavi et al. 379/97

4,922,523 A 5/1990 Hashimoto 379/96
4,975,944 A 12/1990 Cho 379/209
4,995,074 A 2/1991 Goldnan et al. 379/97
5,005,011 A 4/1991 Perlman et al. 340/728
5,095,494 A 3/1992 Takahashi et al. 375/10
5,220,420 A 6/1993 Hoarty et al. 358/86
5,241,587 A 8/1993 Horton et al. 379/92
5,263,084 A 11/1993 Chaput et al. 379/215
5,287,401 A 2/1994 Lin 379/98
5,299,307 A 3/1994 Young 395/161

(List continued on next page.)

OTHER PUBLICATIONS

Rosoff, Matt, Review: "Gateway Destination PC," c/net
inc., 2 pages, Feb. 19, 1996.

Seidman, Robert, Article: "What Larry and Lou Know (That
You Don't)," c/net inc., 2 pages, Jan. 29, 1996.

Stellin, Susan, Article: "The \$500 Web Box: Less is More?"
c/net inc., 2 pages, 1996.

(List continued on next page.)

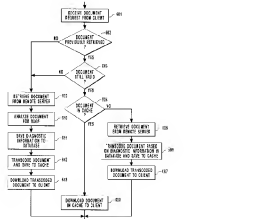
Primary Examiner—Bharat Barot

(74) Attorney, Agent, or Firm—Workman, Nydegger

(57) **ABSTRACT**

A method is described of providing a document to a client
coupled to a server. The server functions as a proxy on
behalf of the client for purposes of accessing a remote
server. In the method, a document is retrieved from the
remote server in response to a request from the client. The
document includes data to be used by the client in generating
a display. The proxying server alters (i.e., transcodes) the
data in the document to form a transcoded document. The
transcoded document is then transmitted to the client. The
proxying server transcodes the data in the document in order
to perform at least one of the following functions: (1)
matching decompression requirements at the client; (2)
converting the document into a format compatible for the
client; (3) reducing latency experienced by the client; and
(4) altering the document to fit into smaller memory space.

31 Claims, 12 Drawing Sheets



U.S. PATENT DOCUMENTS

5,325,423 A	6/1994	Lewis	379/60
5,329,619 A	7/1994	Page et al.	395/200
5,341,293 A	8/1994	Verletney et al.	364/419.17
5,369,688 A	11/1994	Tsukamoto et al.	379/100
5,469,540 A	11/1995	Powers, III et al.	395/158
5,488,411 A	1/1996	Fewis	348/8
5,490,208 A	2/1996	Remillard	379/96
5,530,852 A	6/1996	Meske, Jr. et al.	395/600
5,538,255 A	7/1996	Barker	463/41
5,558,339 A	9/1996	Periman	463/42
5,561,709 A	10/1996	Remillard	379/96
5,564,001 A	10/1996	Lewis	395/154
5,572,643 A	11/1996	Judson	395/793
5,586,257 A	12/1996	Periman	463/42
5,586,260 A	12/1996	Hu	395/200.2
5,612,730 A	3/1997	Lewis	348/8
5,623,600 A	4/1997	Ji et al.	395/187.01
5,654,886 A	8/1997	Zereski, Jr. et al.	364/420
5,657,390 A	8/1997	Elgarni et al.	380/49
5,657,450 A	8/1997	Rao et al.	395/610

5,678,041 A	10/1997	Baker et al.	395/609
5,727,159 A	3/1998	Kikinis	395/200.76
5,842,216 A	11/1998	Anderson et al.	707/203
5,889,955 A	3/1999	Shinozaki et al.	395/200.54
5,918,013 A	* 6/1999	Mighdoli et al.	709/217
5,978,817 A	11/1999	Gianandrea et al.	707/501
5,996,022 A	* 11/1999	Krueger et al.	709/247
6,018,619 A	1/2000	Allard et al.	395/200.54
6,226,412 B1	5/2001	Schwab	382/232

OTHER PUBLICATIONS

Administrator's Guide, Netscape Proxy Server Version 2.0, Netscape Communications Corporation, pp. 19-20, 1996.
 Chankhuthod, Anawat, et al., "A Hierarchical Internet Object Cache," 1996 USEWIX Technical Conference (6 pages).
 Farrow, Rik, "Securing the Web: fire walls, proxy servers, and data driven attacks," InfoWorld, Jun. 19, 1995, vol. 7, No. 25, pp. 103-104.

* cited by examiner

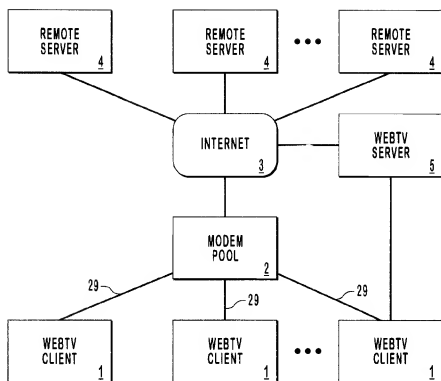


FIG. 1

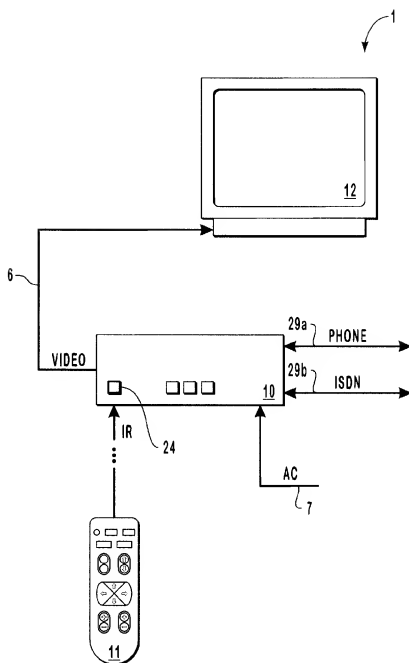


FIG. 2

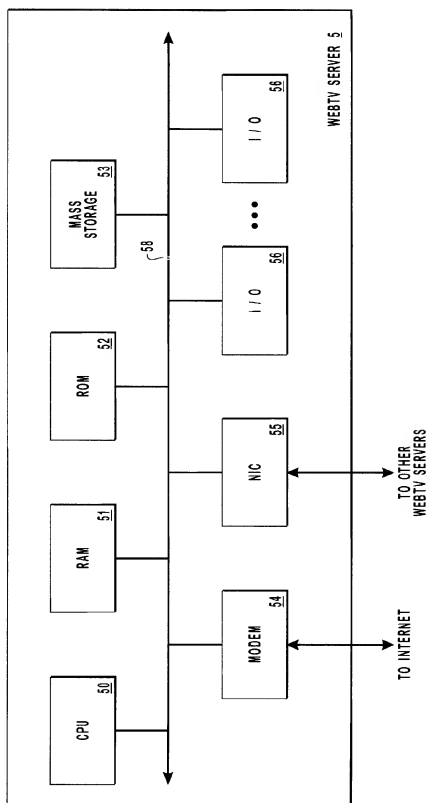


FIG. 3

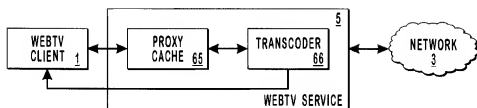


FIG. 4A

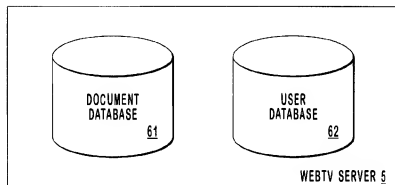


FIG. 4B

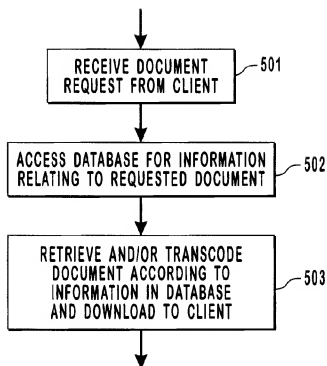


FIG. 5

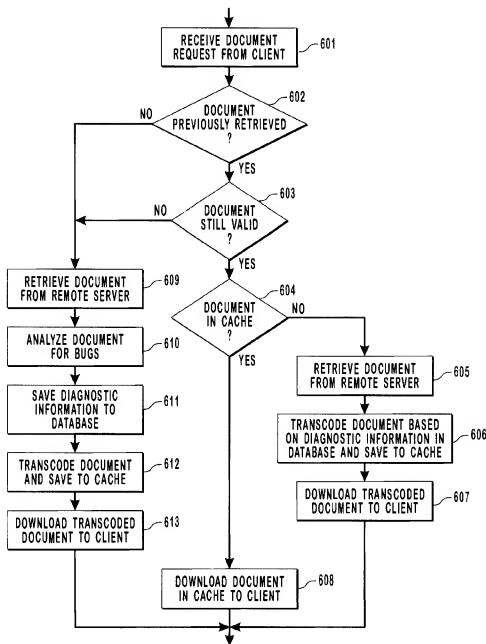


FIG. 6

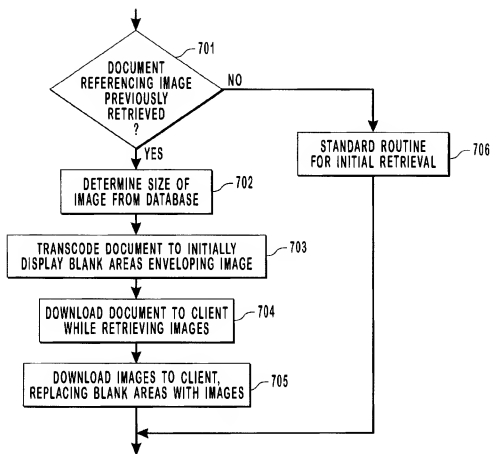


FIG. 7

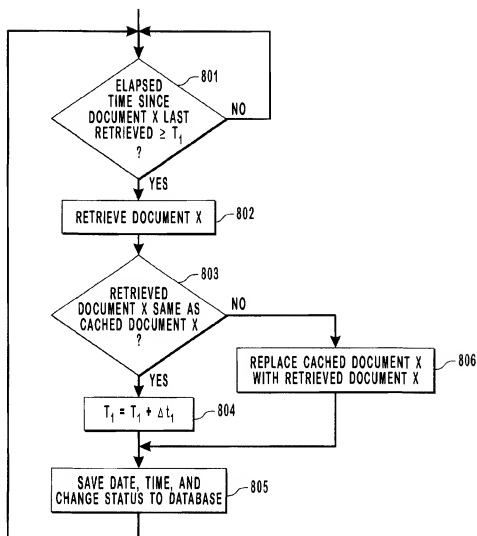


FIG. 8

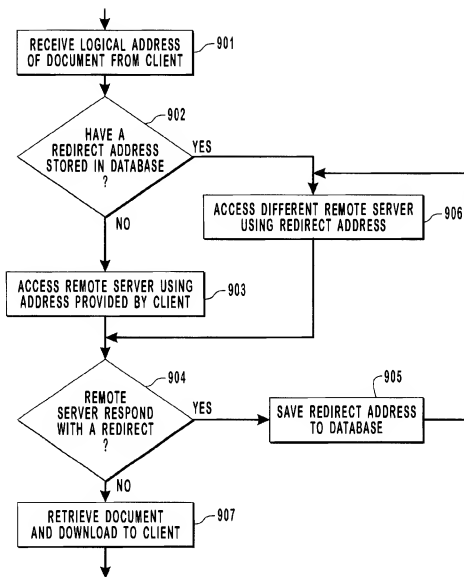


FIG. 9

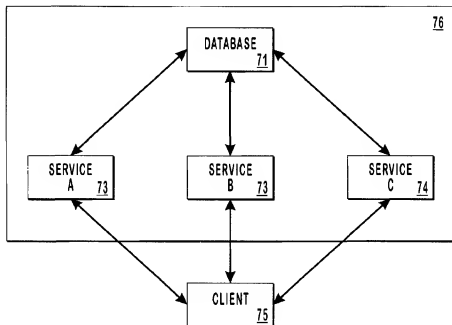


FIG. 10
(PRIOR ART)

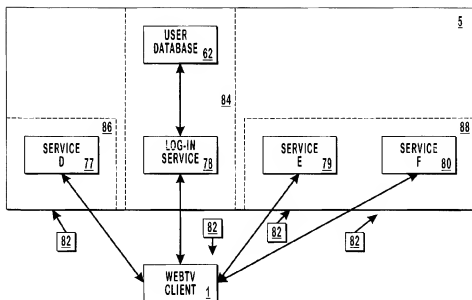


FIG. 11

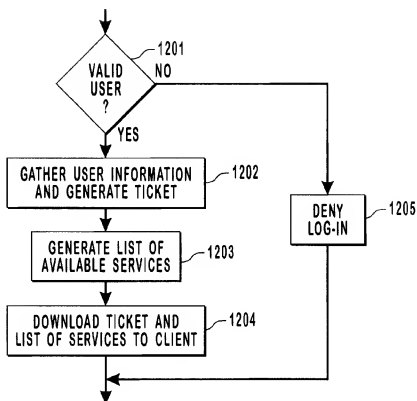


FIG. 12

1

METHOD OF TRANSCODING DOCUMENTS IN A NETWORK ENVIRONMENT USING A PROXY SERVER

RELATED APPLICATION

This is a continuation of U.S. patent application Ser. No. 09/343,067, entitled "Method of Transcoding Documents in a Network Environment Using a Proxy Server," filed Jun. 29, 1999, now abandoned which is a continuation of U.S. application Ser. No. 08/656,924 entitled "Method of Transcoding Documents in a Network Environment Using a Proxy Server," filed Jun. 3, 1996, now U.S. Pat. No. 5,918, 013 both of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. The Field of the Invention

The present invention pertains to the field of client-server computer networking. More particularly, the present invention relates to a method of transcoding documents in a network environment using a proxy server.

2. The Prior State of the Art

The number of people using personal computers has increased substantially in recent years, and along with this increase has come an explosion in the use of the Internet. One particular aspect of the Internet which has gained widespread use is the World-Wide Web ("the Web"). The Web is a collection of formatted hypertext pages located on numerous computers around the world that are logically connected by the Internet. Advances in network technology and software providing user interfaces to the Web ("Web browsers") have made the Web accessible to a large segment of the population. However, despite the growth in the development and use of the Web, many people are still unable to take advantage of this important resource.

Access to the Web has been limited thus far mostly to people who have access to a personal computer. However, many people cannot afford the cost of even a relatively inexpensive personal computer, while others are either unable or unwilling to learn the basic computer skills that are required to access the Web. Furthermore, Web browsers in the prior art generally do not provide the degree of user-friendliness desired by some people, and many computer novices do not have the patience to learn how to use the software. Therefore, it would be desirable to provide an inexpensive means by which a person can access the Web without the use of a personal computer. In particular, it would be desirable for a person to be able to access the Web pages using an ordinary television set and a remote control, so that the person feels more as if he or she is simply changing television channels, rather than utilizing a complex computer network.

Prior art Web technology also has other significant limitations which can make a person's experience unpleasant when browsing the Web. Web documents are commonly written in HTML (Hypertext Mark-up Language). HTML documents sometimes contain bugs (errors) or have features that are not recognized by certain Web browsers. These bugs or quirks in a document can cause a Web browser to fail. Thus, what is needed is a means for reducing the frequency with which client systems fail due to bugs or quirks in HTML documents.

Another problem associated with browsing the Web is latency. People commonly experience long, frustrating delays when browsing the Web. It is not unusual for a person to have to wait minutes after selecting a hypertext link for a

2

Web page to be completely downloaded to his computer and displayed on his computer screen. There are many possible causes for latency, such as heavy communications traffic on the Internet and slow response of remote servers. Latency can also be caused by Web pages including images. One reason for this effect is that, when an HTML document references an image, it takes time to retrieve the image itself after the referencing document has been retrieved. Another reason is that, in the prior art, if the referencing document does not specify the size of the image, the client system generally cannot display the Web page until the image itself has been retrieved. Numerous other sources of latency exist with respect to the Web. Therefore, what is needed is a means for reducing such latency, to eliminate some of the frustration which typically has been associated with browsing the Web.

Security is another concern associated with the Internet. Internet service providers (ISPs) generally maintain certain information about each customer in a database. This information may include information which a customer may not wish to become publicly known, such as social security numbers and credit card numbers. Maintaining the confidentiality of this information in a system that is connected to an expensive publicly-accessible computer network like the Internet can be problematic. Further, the problem can be aggravated by the fact that an ISP often provides numerous different services, each of which has access to this database. Allowing access to the database by many different entities creates many opportunities for security breaches to occur. Therefore, what is needed is a way to improve the security of confidential customer information in a server system coupled to the Internet.

SUMMARY AND OBJECTS OF THE INVENTION

A method is described of providing a document to a client coupled to a server. The server functions as a proxy on behalf of the client for purposes of accessing a remote server. In the method, a document is retrieved from the remote server in response to a request from the client. The document includes data to be used by the client in generating a display. The proxying server alters (i.e., transcodes) the data in the document to form a transcoded document. The transcoded document is then transmitted to the client.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follows. The proxying server transcodes the data in the document in order to perform at least one of the following functions: (1) matching decompression requirements at the client; (2) converting the document into a format compatible for the client; (3) reducing latency experienced by the client; and (4) altering the document to fit into smaller memory space.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follow.

BRIEF DESCRIPTION OF THE DRAWINGS

In order that the manner in which the above-recited and other advantages and objects of the invention are obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered limiting of its scope, the invention will be

3

described and explained with additional specificity and detail through the use of the accompanying drawings in which:

FIG. 1 illustrates several clients connected to a proxying server in a network;

FIG. 2 illustrates a client according to the present invention;

FIG. 3 is a block diagram of a server according to the present invention;

FIG. 4A illustrates a server including a proxy cache and a transcoder;

FIG. 4B illustrates databases used in a server according to the present invention;

FIG. 5 is a flow diagram illustrating a routine for transcoding a document retrieved from a remote server using data stored in a persistent database;

FIG. 6 is a flow diagram illustrating a routine for transcoding an HTML document for purposes of eliminating bugs or undesirable features;

FIG. 7 is a flow diagram illustrating a routine for reducing latency when downloading a document referencing an image to a client;

FIG. 8 is a flow diagram illustrating a routine for updating documents stored in the proxy cache using data stored in a persistent database;

FIG. 9 is a flow diagram illustrating a routine used by a server for retrieving documents from another remote server;

FIG. 10 is a block diagram of a prior art server system showing a relationship between various services and a database;

FIG. 11 is a block diagram of a server system according to the present invention showing a relationship between various services and a user database; and

FIG. 12 is a flow diagram illustrating a routine used by a server for regulating access to various services provided by the server.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

A method and apparatus are described for providing proxying and transcoding of documents in a network. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be evident, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

The present invention includes various steps, which will be described below. The steps can be embodied in machine-executable instructions, which can be used to cause a general-purpose or special-purpose processor programmed with the instructions to perform the steps. Alternatively, the steps of the present invention might be performed by specific hardware components that contain hardwired logic for performing the steps, or by any combination of programmed computer components and custom hardware components.

I. System Overview

The present invention is included in a system, known as WebTV™, for providing a user with access to the Internet. A user of a WebTV™ client generally accesses a WebTV™ server via a direct-dial telephone (POTS, for "plain old

4

telephone service"), ISDN (Integrated Services Digital Network), or other similar connection, in order to browse the Web, send and receive electronic mail (e-mail), and use various other WebTV™ network services. The WebTV™ network services are provided by WebTV™ servers using software residing within the WebTV™ servers in conjunction with software residing within a WebTV™ client.

FIG. 1 illustrates a basic configuration of the WebTV™ network according to one embodiment. A number of WebTV™ clients 1 are coupled to a modem pool 2 via direct-dial, bi-directional data connections 29, which may be telephone (POTS, i.e., "plain old telephone service"), ISDN (Integrated Services Digital Network), or any other similar type of connection. The modem pool 2 is coupled typically through a router, such as that conventionally known in the art, to a number of remote servers 4 via a conventional network infrastructure 3, such as the Internet. The WebTV™ system also includes a WebTV™ server 5, which specifically supports the WebTV™ clients 1. The WebTV™ clients 1 each have a connection to the WebTV™ server 5 either directly or through the modem pool 2 and the Internet 3. Note that the modem pool 2 is a conventional modem pool, such as those found today throughout the world providing access to the Internet and private networks.

Note that in this description, in order to facilitate explanation the WebTV™ server 5 is generally discussed as if it were a single device, and functions provided by the WebTV™ services are generally discussed as being performed by such single device. However, the WebTV™ server 5 may actually comprise multiple physical and logical devices connected in a distributed architecture, and the various functions discussed below which are provided by the WebTV™ services may actually be distributed among multiple WebTV™ server devices.

II. Client System

FIG. 2 illustrates a WebTV™ client 1. The WebTV™ client 1 includes an electronics unit 10 (hereinafter referred to as "the WebTV™ box 10"), an ordinary television set 12, and a remote control 11. In an alternative embodiment of the present invention, the WebTV™ box 10 is built into the television set 12 as an integral unit. The WebTV™ box 10 includes hardware and software for providing the user with a graphical user interface, by which the user can access the WebTV™ network services, browse the Web, send e-mail, and otherwise access the Internet.

The WebTV™ client 1 uses the television set 12 as a display device. The WebTV™ box 10 is coupled to the television set 12 by a video link 6. The video link 6 is an RF (radio frequency), S-video, composite video, or other equivalent form of video link. In the preferred embodiment, the client 1 includes both a standard modem and an ISDN modem, such that the communication link 29 between the WebTV™ box 10 and the server 5 can be either a telephone (POTS) connection 29a or an ISDN connection 29b. The WebTV™ box 10 receives power through a power line 7.

Remote control 11 is operated by the user in order to control the WebTV™ client 1 in browsing the Web, sending e-mail, and performing other Internet-related functions. The WebTV™ box 10 receives commands from remote control 11 via an infrared (IR) communication link. In alternative embodiments, the link between the remote control 11 and the WebTV™ box 10 may be RF or any equivalent mode of transmission.

III. Server System

The WebTV™ server 5 generally includes one or more computer systems generally having the architecture illus-

5

trated in FIG. 3. It should be noted that the illustrated architecture is only exemplary, the present invention is not constrained to this particular architecture. The illustrated architecture includes a central processing unit (CPU) 50, random access memory (RAM) 51, read-only memory (ROM) 52, a mass storage device 53, a modem 54, a network interface card (NIC) 55, and various other input/output (I/O) devices 56. Mass storage device 53 includes a magnetic, optical, or other equivalent storage medium. I/O devices 56 may include any or all of devices such as a display monitor, keyboard, cursor control device, etc. Modem 54 is used to communicate data to and from remote servers 4 via the Internet.

As noted above, the WebTV™ server 5 may actually comprise multiple physical and logical devices connected in a distributed architecture. Accordingly, NIC 55 is used to provide data communication with other devices that are part of the WebTV™ services. Modem 54 may also be used to communicate with other devices that are part of the WebTV™ services and which are not located in close geographic proximity to the illustrated device.

According to the present invention, the WebTV™ server 5 acts as a proxy in providing the WebTV™ client 1 with access to the Web and other WebTV™ services. More specifically, WebTV™ server 5 functions as a "caching proxy." FIG. 4A illustrates the caching feature of the WebTV™ server 5. In FIG. 4A, the WebTV™ server 5 is functionally located between the WebTV™ client 1 and the Internet infrastructure 3. The WebTV™ server 5 includes a proxy cache 65 which is functionally coupled to the WebTV™ client 1. The proxy cache 65 is used for temporary storage of Web documents, images, and other information which is frequently used by either the WebTV™ client 1 or the WebTV™ server 5.

A document transcoder 66 is functionally coupled between the proxy cache 65 and the Internet infrastructure 3. The document transcoder 66 includes software which is used to automatically revise the code of Web documents retrieved from the remote servers 4, for purposes which are described below.

The WebTV™ service provides a document database 61 and a user database 62, as illustrated in FIG. 4B. The user database 62 contains information that is used to control certain features relating to access privileges and capabilities of the user of the client 1. This information is used to regulate initial access to the WebTV™ service, as well as to regulate access to the individual services provided by the WebTV™ system, as will be described below. The document database 61 is a persistent database which stores certain diagnostic and historical information about each document and image retrieved by the server 5, as is now described.

A. Document Database

The basic purpose of the document database 61 is that, after a document has once been retrieved by the server 5, the stored information can be used by the server 5 to speed up processing and downloading of that document in response to all future requests for that document. In addition, the transcoding functions and various other functions of the WebTV™ service are facilitated by making use of the information stored in the document database 61, as will be described below.

Referring now to FIG. 5, the server 5 initially receives a document request from a client 1 (step 501). The document request will generally result from the user of the client 1

6

activating a hypertext anchor (link) on a Web page. The act of activating a hypertext anchor may consist of clicking on underlined text in a displayed Web page using a mouse, for example. The document request will typically (but not always) include the URL (Uniform Resource Locator) or other address of the selected anchor. Upon receiving the document request, the server 5 optionally accesses the document database 61 to retrieve stored information relating to the requested document (step 502). It should be noted that the document database 61 is not necessarily accessed in every case. The information retrieved from the document database 61 is used by the server 5 for determining, among other things, how long a requested document has been cached and/or whether the document is still valid. The criteria for determining validity of the stored document are discussed below. The server 5 retrieves the document from the cache 65 if the stored document is valid; otherwise, the server 5 retrieves the document from the appropriate remote server 4 (step 503). The server 5 automatically transcodes the document as necessary based on the information stored in the document database 61 (step 503). The transcoding functions are discussed further below.

The document database 61 includes certain historical and diagnostic information for every Web page that is accessed at any time by a WebTV™ client 1. As is well known, a Web page may correspond to a document written in a language such as HTML (Hypertext Mark-Up Language), VRML (Virtual Reality Modelling Language), or another suitable language. Alternatively, a Web page may represent an image, or a document which references one or more images. According to the present invention, once a document or image is retrieved by the WebTV™ server 5 from a remote server 4 for the first time, detailed information on this document or image is stored permanently in the document database 61. More specifically, for every Web page that is retrieved from a remote server 4, any or all of the following data are stored in the document database 61:

- 1) information identifying bugs (errors) or quirks in the Web page, or undesirable effects caused when the Web page is displayed by a client 1;
- 2) relevant bug-finding algorithms;
- 3) the date and time the Web page was last retrieved;
- 4) the date and time the Web page was most recently altered by the author;
- 5) a checksum for determining whether the Web page has been altered;
- 6) the size of the Web page (in terms of memory);
- 7) the type of Web page (e.g., HTML document, image, etc.);
- 8) a list of hypertext anchors (links) in the Web page and corresponding URLs;
- 9) a list of the most popular anchors based on the number of "hits" (requests from a client 1);
- 10) a list of related Web pages which can be prefetched;
- 11) whether the Web page has been redirected to another remote server 4;
- 12) a redirect address (if appropriate);
- 13) whether the redirect (if any) is temporary or permanent, and if permanent, the duration of the redirect;
- 14) if the Web page is an image, the size of the image in terms of both physical dimensions and memory space;
- 15) the sizes of in-line images (images displayed in text) referenced by the document defining the Web page;

7

- 16) the size of the largest image referenced by the document;
- 17) information identifying any image maps in the Web page;
- 18) whether to resize any images corresponding to the Web page;
- 19) an indication of any forms or tables in the Web page;
- 20) any unknown protocols;
- 21) any links to "dead" Web pages (i.e., pages which are no longer active);
- 22) the latency and throughput of the remote server 4 on which the Web page is located;
- 23) the character set of the document;
- 24) the vendor of the remote server 4 on which the Web page is located;
- 25) the geographic location of the remote server 4 on which the Web page is located;
- 26) the number of other Web pages which reference the subject Web page;
- 27) the compression algorithm used by the image or document;
- 28) the compression algorithm chosen by the transcoder;
- 29) a value indicating the popularity of the Web page based on the number of hits by clients; and
- 30) a value indicating the popularity of other Web pages which reference the subject Web page.

B. Transcoding

As mentioned above, the WebTV™ services provide a transcoder 66, which is used to rewrite certain portions of the code in an HTML document for various purposes. These purposes include: (1) correcting bugs in documents; (2) correcting undesirable effects which occur when a document is displayed by the client 1; (3) improving the efficiency of transmission of documents from the server 5 to the client 1; (4) matching hardware decompression technology within the client 1; (5) resizing images to fit on the television set 12; (6) converting documents into other formats to provide compatibility; (7) reducing latency experienced by a client 1 when displaying a Web page with in-line images (images displayed in text); and, (8) altering documents to fit into smaller memory spaces.

There are three transcoding modes used by the transcoder 66: (1) streaming, (2) buffered, and (3) deferred. Streaming transcoding refers to the transcoding of documents on a line-by-line basis as they are retrieved from a remote server 4 and downloaded to the client 1 (i.e., transcoding "on the fly"). Some documents, however, must first be buffered in the WebTV™ server 5 before transcoding and downloading them to the client 1. A document may need to be buffered before transmitting it to the client 1 if the type of changes to be made can only be made after the entire document has been retrieved from the remote server 4. Because the process of retrieving and downloading a document to the client 1 increases latency and decreases throughput, it is not desirable to buffer all documents. Therefore, the transcoder 66 accesses and uses information in the document database 61 relating to the requested document to first determine whether a requested document must be buffered for purposes of transcoding, before the document is retrieved from the remote server 4.

In the deferred mode, transcoding is deferred until after a requested document has been downloaded to a client 1. The deferred mode therefore reduces latency experienced by the

8

client 1 in receiving the document. Transcoding may be performed immediately after downloading or any time thereafter. For example, it may be convenient to perform transcoding during periods of low usage of WebTV™ services, such as at night. This mode is useful for certain types of transcoding which are not mandatory.

1. Transcoding for Bugs and Quirks

One characteristic of some prior art Web browsers is that they may experience failures ("crashes") because of bugs or unexpected features ("quirks") that are present in a Web document. Alternatively, quirks in a document may cause an undesirable result, even though the client does not crash. Therefore, the transcoding feature of the present invention provides a means for correcting certain bugs and quirks in a Web document. To be corrected by the transcoder 66, bugs and quirks must be identifiable by software running on the server 5. Consequently, the transcoder 66 will generally only correct conditions which have been previously discovered, such as those discovered during testing or reported by users. Once a bug or quirk is discovered, however, algorithms are added to the transcoder 66 to both detect the bug or quirk in the future in any Web document and to automatically correct it.

There are countless possibilities of bugs or quirks which might be encountered in a Web document. Therefore, no attempt will be made herein to provide an exhaustive list. Nonetheless, some examples may be useful at this point. Consider, for example, an HTML document that is downloaded from a remote server 4 and which contains a table having a width specified in the document as "0.". This condition might cause a failure if the client were to attempt to display the document as written. This situation therefore, can be detected and corrected by the transcoder 66. Another example is a quirk in the document which causes quotations to be terminated with too many quotation marks. Once the quirk is first detected and an algorithm is written to recognize it, the transcoder 66 can automatically correct the quirk in any document.

If a given Web document has previously been retrieved by the server 5, there will be information regarding that document available in the document database 61 as described above. The information regarding this document will include whether or not the document included any bugs or quirks that required transcoding when the document was previously retrieved. The transcoder 66 utilizes this information to determine whether (1) the document is free of bugs and quirks, (2) the document has bugs or quirks which can be remedied by transcoding on the fly, or (3) the document has bugs or quirks which cannot be corrected on the fly (i.e., buffering is required).

FIG. 6 illustrates a routine for transcoding a Web document for purposes of eliminating bugs and quirks. Initially, the server 5 receives a document request from the client 1 (step 601). Next, the document database 61 is accessed to determine whether or not the requested document has been previously retrieved (step 602). If the document has not been previously retrieved, then the server 5 retrieves the document from the remote server 4 (step 609). Next, the retrieved document is analyzed for the presence of bugs or unusual conditions (step 610). Various diagnostic information is then stored in the document database 61 as a result of the analysis to note any bugs or quirks that were found (step 611). If any bugs or quirks were found which can be corrected by the transcoder 66, the document is then transcoded and saved to the proxy cache 65 (step 612). The transcoded document is

9

then downloaded to the client 1 (step 613). It should be noted that transcoding can be deferred until after the document has been downloaded, as described above; hence, the sequence of FIG. 6 is illustrative only.

If (in step 602) the requested document had been previously retrieved, then it is determined whether the requested document is still valid (step 603) and whether the document is present in the proxy cache 65 (step 604). If the document is no longer valid, then the document is retrieved from the remote server 4, analyzed for bugs and quirks, transcoded as required, and then downloaded to the client 1 as described above (steps 609-613). Methods for determining validity of a document are discussed below. If the document is still valid (step 603) and the document is present in the cache 65, the document is downloaded to the client 1 in its current form (as it is stored in the cache), since it has already been transcoded (step 608).

The document, however, may be valid but not present in the cache. This may be the case, for example, if the document has not been requested recently and the cache 65 has become too full to retain the requested document. In that case, the document is retrieved again from the remote server 4 (step 605) and then transcoded on the basis of the previously-acquired diagnostic information stored within the database 61 for that document. The document is then saved to the cache 65 (step 606). Note that because the document is still valid, it is assumed that the diagnostic information stored in the document database 61 for that document is still valid and that the transcoding can be performed on the basis of that information. Accordingly, once the document is transcoded, the transcoded document is downloaded to the client 1 (step 607). Again, note that transcoding can be deferred until after the document has been downloaded in some cases.

The validity of the requested document can be determined based on various different criteria. For example, some HTML documents specify a date on which the document was created, a length of time for which the document will be valid, or both. The validity determination can be based upon such information. For example, a document which specifies only the date of creation can be automatically deemed invalid after a predetermined period of time has passed.

Alternatively, validity can be based upon the popularity of the requested document. "Popularity" can be quantified based upon the number of hits for that document, which is tracked in the document database 61. For example, it might be prudent to simply assign a relatively short period of validity to a document which is very popular and a longer period of validity to a document which is less popular.

Another alternative basis for the validity of a document is the observed rate of change of the document. Again, data in the persistent document database 61 can be used. That is, because the document database 61 stores the date and time on which the document was last observed to change, the server 5 can approximate how often the document actually changes. A document or image which is observed to change frequently (e.g., a weather map or a news page) can be assigned a relatively short period of validity. It will be recognized that numerous other ways of determining validity are possible.

2. Transcoding to Reduce Latency

Another purpose for transcoding is to allow documents requested by a client 1 to be displayed by the client 1 more rapidly. Many HTML documents contain references to "in-line" images, or images that will be displayed in text in a

10

Web page. The normal process used in the prior art to display a Web page having in-line images is that the HTML document referencing the image is first downloaded to the client, followed by the client's requesting the referenced image. The referenced image is then retrieved from the remote server on which it is located and downloaded to the client. One problem associated with the prior art, however, is that the speed with which a complete Web page can be displayed to the user is often limited by the time it takes to retrieve in-line images. One reason for this is that it simply takes time to retrieve the image itself after the referencing document has been retrieved. Another reason is that, in the prior art, if the referencing document does not specify the size of the image, the Web page generally cannot be displayed until the image itself has been retrieved. The present invention overcomes these limitations.

According to the present invention, information stored in the document database 61 regarding the in-line images is used to transcode the referencing document in order to reduce latency in displaying the Web page. Once any document which references an in-line image is initially retrieved by the server 5, the fact that the document references an in-line image is stored in the document database 61. In addition, the size of the image is determined, either from the document (if specified) or from the image itself, and then stored in the document database 61. Consequently, for documents which do not specify the size of their in-line images, the size information stored in the database 61 is then used the next time the document is requested in order to reduce latency in downloading and displaying the Web page.

Refer now to FIG. 7, which illustrates a routine for reducing latency when downloading a document referencing an image to a client 1. Assume that a client 1 sends a request to the server 5 for an HTML document containing a reference to an in-line image. Assume further that the size of the image is not specified in the document itself. Initially, the server 5 determines whether that document has been previously retrieved (step 701). If not, the standard initial retrieval and transcoding procedure is followed (step 706), as described in connection with FIG. 6. If, however, the document has been previously retrieved, then the transcoder 66 accesses the size information stored in the document database 61 for the in-line image (step 702). Based on this size information, the HTML document is transcoded such that, when the Web page is initially displayed by the client 1, the area in which the image belongs is replaced by a blank region enveloping the shape of the image (step 703). Thus, any in-line image referenced by a document is displayed initially as a blank region. Consequently, the client 1 can immediately display the Web page corresponding to the HTML document even before the referenced image has been retrieved or downloaded (i.e., even before the size of the image is known to the client 1).

As the transcoded HTML document is downloaded to the client, the image is retrieved from the appropriate remote server 4 (step 704). Once the image is retrieved from the remote server 4 and downloaded to the client 1, the client 1 replaces the blank area in the Web page with the actual image (step 705).

3. Transcoding to Display Web Pages on a Television

As noted above, the client 1 utilizes an ordinary television set 12 as a display device. However, images in Web pages are generally formatted for display on a computer monitor, not a television set. Consequently, the transcoding function

11

of the present invention is used to resize images for display on the television set 12. This includes resealing images as necessary to avoid truncation when displayed on the television set 12.

It should be noted that prior art Web browsers which operate on computer monitors typically use resizable windows. Hence, the size of the visible region varies from client to client. However, because the web browser used by the WebTV™ client 1 is specifically designed for display on a television set, the present invention allows documents and images to be formatted when they are cached.

4. Transcoding for Transmission Efficiency

Documents retrieved by the server 5 are also transcoded to improve transmission efficiency. In particular, documents can be transcoded in order to reduce high frequency components in order to reduce interface flicker when they are displayed on a television set. Various methods for coding software or hardware to reduce perceptual interface flicker are described in co-pending U.S. patent application Ser. No. 08/656,923, filed on Jun. 3, 1996.

Documents can also be transcoded in order to lower the resolution of the displayed Web page. Reducing the resolution is desirable, because images formatted for computer systems will generally have a higher resolution than the NTSC (National Television Standards Committee) video format used by conventional television sets. Since the NTSC video does not have the bandwidth to reproduce the resolution of computer-formatted images, the bandwidth consumed in transmitting images to the client 1 at such a high resolution would be wasted.

5. Other Uses for Transcoding

Transcoding is also used by the present invention to recode a document using new formats into older, compatible formats. Images are often displayed in the JPEG (Joint Picture Experts Group) format or the GIF image format. JPEG often consumes less bandwidth than GIF, however. Consequently, images which are retrieved in GIF format are sometimes transcoded into JPEG format. Methods for generally converting images between GIF and JPEG formats are well known.

Other uses for transcoding include transcoding audio files. For example, audio may be transcoded into different formats in order to achieve a desired balance between memory usage, sound quality, and data transfer rate. In addition, audio may be transcoded from a file format (e.g., an "AU" file) to a streaming format (e.g., MPEG 1 audio). Yet another use of audio transcoding is the transcoding of MIDI (Musical Instrument Digital Interface) data to streaming variants of MIDI.

Additionally, documents or images requiring a large amount of memory (e.g., long lists) can be transcoded in order to consume less memory space in the client 1. This may involve, for example, separating a large document or image into multiple sections. For example, the server 5 can insert tags at appropriate locations in the original document so that the document appears to the client 1 as multiple Web pages. Hence, while viewing a given page representing a portion of the original document, the user can view the next page (i.e., the next portion of the original document) by activating a button on the screen as if it were an ordinary hypertext anchor.

C. Proxying

As noted above, the server 5 functions as a proxy on behalf of the client 1 for purposes of accessing the Web. The

12

document database 61 is used in various ways to facilitate this proxy role, as will now be described.

1. Updating Cached Documents

It is desirable to store frequently-requested HTML documents and images in the proxy cache 65 to further reduce latency in providing Web pages to the client 1. However, because some documents and images change over time, documents in the cache 65 will not be valid indefinitely, as mentioned above. A weather map or a news-related Web page, for example, are likely to be updated quite frequently. Consequently, it is desirable for the server 5 to have the ability to estimate the frequency with which documents change, in order to determine how long a document can safely remain within the proxy cache 65 without being updated.

The persistent database 65 is used to store the date and time of the last several fetches of each document and image retrieved from a remote server 4, along with an indication of any changes that were detected, if any. A document or image which has been stored in the cache 65 is then retrieved on a periodic basis to determine if it has been changed. Change status information indicating whether the document has changed since the previous fetch is then stored in the document database 61. If no changes are detected, then the time interval between fetches of this document is increased. If the document has changed, the time interval is maintained or decreased. As a result, items in the cache 65 which change frequently will be automatically updated at frequent intervals, whereas documents which do not change often will be replaced in the cache less frequently.

FIG. 8 illustrates a routine for updating documents stored in the proxy cache 65 using data stored in the document database 61. Assume a document X has been stored in the proxy cache 65. Document X remains in the cache 65 until a predetermined update period T_1 expires (step 801). Upon the expiration of the update period T_1 , the document X is again retrieved from the appropriate remote server 4 (step 802). The newly-retrieved document X is then compared to the cached version of document X (step 803). If the document has changed, then the cached version of document X is replaced with the newly-retrieved version of document X (step 806). If not, then the update period T_1 is increased according to a predetermined time increment ΔT_1 (step 804). In any case, the date and time and the change status of document X is saved to the document database 61 (step 805).

2. Document and Image Prefetching

The document database 61 is also used by the server 5 to store prefetching information relating to documents and images. In particular, the database stores, for each document that has been retrieved, a list of images referenced by the document, if any, and their locations. Consequently, the next time a document is requested by a client 1, the images can be immediately retrieved by the server 5 (from the cache 65, if available, or from the remote server 4), even before the client 1 requests them. This procedure improves the speed with which requested Web pages are downloaded to the client.

The document database 61 is also used to facilitate a process referred to as "server-advised client prefetching." Server-advised client prefetching allows the server 5 to inform the client 1 of documents or images which are popular to allow the client 1 to perform the prefetching. In particular, for any given document, a list is maintained in the

13

server 5 of the most popular hypertext anchors in that document (i.e., those which have previously received a large number of hits). When that document is requested by the client 1, the server 5 provides the client 1 with an indication of these popular links.

3. Redirects

Web pages are sometimes forwarded from the remote server on which they are initially placed to a different location. Under the HTTP (Hypertext Transport Protocol), such forwarding is sometimes referred to as a "redirect." When an HTML document is initially stored on one remote server and then later transferred to another remote server, the first remote server will provide, in response to a request for that document, an indication that the document has been transferred to a new remote server. This indication generally includes a forwarding address ("redirect address"), which is generally a URL.

In the prior art, when a computer requesting a Web page receives a redirect, it must then submit a new request to the redirect address. Having to submit a second request and wait for a second response consumes time and increases overall latency. Consequently, the present invention uses the document database 61 to store any redirect address for each document or image. Any time a redirected document is requested, the server 5 automatically accesses the redirect address to retrieve the document. The document or image is provided to the client 1 based on only a single request from the client 1. The change in location of the redirected document or image remains completely transparent to the client 1.

FIG. 9 illustrates a routine performed by the server 5 in accessing documents which may have been forwarded to a new remote server. Initially, the server 5 receives a request for a document, which generally includes an address (step 901). The server 5 then accesses the document database 61 to determine whether there is a redirect address for the requested document (step 902). If there is no redirect address, then the server 5 accesses a remote server 4 based on the address provided in the document request from the client 1 (step 903). Assuming that the remote server 4 does not respond to the server 5 with a redirect (step 904), the document is retrieved and downloaded to the client 1 by the server 5 (step 907). If, however, a redirect address was stored in the document database 61 (step 902), then the server 5 accesses the requested document according to the redirect address (step 906). Or, if the remote server 4 responded with a redirect (step 904), then the server 5 saves the redirect address to the document database 61 (step 905) and accesses the requested document according to the redirect address (step 906).

4. Other Proxy Functions

The document database 61 also stores information relating to the performance of each remote server 4 from which a document is retrieved. This information includes the latency and throughput of the remote server 4. Such information can be valuable in instances where a remote server 4 has a history of responding slowly. For example, when the document is requested, this knowledge can be used by the server 5 to provide a predefined signal to the client 1. The client 1 can, in response to the signal, indicate to the user that a delay is likely and give the user the option of canceling the request.

5. Backoff Mode

Although the server 5 generally operates in the proxy mode, it can also enter a "backoff mode" in which the server

14

5 does not act as a proxy, or the server 5 performs only certain aspects of the normal proxying functions. For example, if the proxy cache 65 is overloaded, then the server 5 can enter a backoff mode in which documents are not cached but are transcoded as required. Alternatively, during times when the server 5 is severely overloaded with network traffic, the server 5 may instruct the client 1 to bypass the server 5 and contact remote servers 4 directly for a specified time or until further notice. Or, the server 5 can enter a flexible backoff mode in which the client 1 will be instructed to contact a remote server 4 directly only for certain Web sites for a limited period of time.

D. Access to WebTV™ Services

The WebTV™ server 5 provides various services to the client 1, such as proxying and electronic mail ("e-mail"). In the prior art, certain difficulties are associated with allowing a client computer access to different services of an Internet service, as will now be explained with reference to FIG. 10.

FIG. 10 illustrates a client-server system according to one prior art embodiment. The server 76 provides various services A, B, and C. The server 76 includes a database 71 for storing information on the user's access privileges to services A, B, and C. The client 75 of the embodiment of FIG. 10 accesses any of services A, B, and C by contacting that service directly. The contacted service then accesses the database 71, which stores the access privileges of the client 75, to determine whether the client 75 should be allowed to access that service. Hence, each service provided by the server 76 requires direct access to the database 71. This architecture results in a large number of accesses being made to the database 71, which is undesirable. In addition, the fact that each service independently has access to the database 71 raises security concerns. Specifically, it can be difficult to isolate sensitive user information. The present invention overcomes such difficulties using a technique which is now described.

1. Tickets Containing Privileges And Capabilities

As shown in FIG. 11, the server 5 provides a number of services D, E, and F 77, 79, and 80, respectively, and a log-in service 78. The log-in service 78 is used specifically to control initial log-on procedures by a client 1. The log-in service 78 has exclusive access to the user database 62 (discussed above with respect to FIG. 4B). The log-in service 78 and the user database 62 are located within a first security zone 84. Service D is located within a second security zone 86, while services E and F are contained within a third security zone 88. Note that the specific arrangement of security zones 84, 86, and 88 with respect to services D, E, and F is illustrative only.

The user database 62 of the present invention stores various information pertaining to each authorized user of a client 1. This information includes account information, a list of the WebTV™ services that are available to the particular user, and certain user preferences. For example, a particular user may not wish his client 1 to be used to access Web pages having adult-oriented subject matter. Consequently, the user would request that his account be filtered to prevent access to such material. This request would then be stored as part of the user data in the user database 62.

With regard to user preferences, the hypertext links selected by a given user can be tracked, and those having the largest number can be stored in the user database 62. The list can then be provided to the client 1 for use in generating a

15

menu screen of the user's favorite Web sites, to allow the user to directly access those Web sites. The list can also be used by the server 5 to analyze the user's interests and to formulate and provide to the user a list of new Web sites which the user is likely to be interested in. The list might be composed by associated key words in Web pages selected by the user with other Web pages.

Referring again to FIG. 11, in response to a log-on request by a client 1, the log-in service 78 consults the user database 62 to determine if access to the server 5 by this particular client 1 is authorized. Assuming access is authorized, the log-in service 78 retrieves certain user information pertaining to this particular client 1 from the user database 62. The log-in service then generates a "ticket" 82, which is an information packet including the retrieved information. The ticket 82 is then provided to the client 1 which requested access.

The ticket 82 includes all information necessary to describe the access privileges of a particular user with respect to all services provided by the server 5. For example, the ticket may include the user name registered to the client 1, the e-mail address assigned to client 1, and any filtering requested by the user with respect to viewing Web sites. Each time the user requests access to one of the services D, E, or F, the client 1 submits a copy of the ticket 82 to that service. The requested service can then determine from the copy of the ticket 82 whether access to that service by that client 1 is authorized and, if so, any important information relating to such access.

None of the services provided by the server 5, other than the log-in service 78, has access to the user database 62. Hence, any security-sensitive information can be isolated within the user database 62 and the log-in service 78. Such isolation allows the individual services provided by the server 5 to be placed within separate "firewalls" (security regions), illustrated as security zones 84, 86, and 88. In addition, this technique greatly reduces the number of accesses required to the user database 62 compared to the prior art embodiment illustrated in FIG. 10.

2. Redundancy of Services and Load Balancing

The present invention also includes certain redundancies in the various services provided by the server 5. In particular, a given service (e.g., e-mail) can be provided by more than one physical or logical device. Each such device is considered a "provider" of that service. If a given provider is overloaded, or if the client 1 is unable to contact that provider, the client 1 can contact any of the other providers of that service. When the server 5 receives a log-in request from a client 1, in addition to generating the above-described ticket 82, the log-in service 78 dynamically generates a list of available WebTV™ services and provides this list to the client 1.

The server 5 can update the list of services used by any client 1 to reflect services becoming unavailable or services coming on-line. Also, the list of services provided to each client 1 can be updated by the server 5 based upon changes in the loading of the server 5, in order to optimize traffic on the server 5. In addition, a client's list of services can be updated by services other than the log-in service 78, such that one service can effectively introduce another service to the client 1. For example, the e-mail service may provide a client 1 with the name, port number and IP of its address book service. Thus, one service can effectively, and securely within the same chain of trust, introduce another service to the client 1.

16

This list of services includes the name of each service, a port number for the provider of each service, and an IP (Internet Protocol) for each service. Different providers of the same service are designated by the same name, but different port numbers and/or IPs. Note that in a standard URL, the protocol is normally specified at the beginning of the URL, such as "HTTP://www. . . ." under the HTTP protocol. However, according to the present invention, the normal protocol designation (i.e., "HTTP") in the URL is replaced with the name of the service, since the port number and IP for each service are known to the client 1. Hence, the client 1 can access any of the redundant providers of a given service using the same URL. This procedure effectively adds a level of indirection to all accesses made to any WebTV™ service and automatically adds redundancy to the proxy service. It should also be noted that separate service names can also refer to the same service.

Assume, for example, that the e-mail service provided by the WebTV™ system is designated by the service name "WTV-mailto." A client 1 can access any provider of this e-mail service using the same URL. The client 1 merely chooses the appropriate port number and IP number to distinguish between providers. If the client 1 is unable to connect to one e-mail provider, it can simply contact the next one in the list.

Thus, at log-in time, a client 1 is provided with both a ticket containing privileges and capabilities as well as a list of service providers, as illustrated in FIG. 12. Initially, the log-in service 78 determines whether the user of client 1 is a valid user (step 1201). If not, log-in is denied (step 1205). If the user is a valid user, then the log-in service 78 gathers user information from the user database 62 and generates a ticket 82 (step 1202). The log-in service 78 also generates the above-described list of services (step 1203). The ticket 82 and the list of services are then downloaded to the client 1 (step 1204).

3. Asynchronous Notification to Clients by Server

Another limitation associated with prior art Internet servers is the inability to provide asynchronous notification information to the client in the absence of a request from the client to do so. It would be desirable, for example, for a server to notify a client on its own initiative when a particular Web page has changed or that a particular service is inaccessible. The server 5 of the present invention provides such capability, and the client 1 is configured to receive and decode such notifications. For example, the client 1 can receive updates of its listing of service providers from the server 5 at various points in time, as already described. Similarly, if a particular service provider becomes unavailable, that fact will be automatically communicated to the client 1. As another example, if e-mail addressed to the user has been received by the server 5, then the server 5 will send a message to the client 1 indicating this fact. The client 1 will then notify the user that e-mail is waiting by a message displayed on the television set 12 or by an LED (light emitting diode) built into the housing of WebTV™ box 10.

Thus, a method and apparatus have been described for providing proxying and transcoding of documents in a network. The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

17

What is claimed and desired to be secured by United States Letters Patent is:

1. In a proxying server coupled to a client and to a remote server, the proxying server operating as a proxy on behalf of the client for accessing the remote server, a method of retrieving and transcoding a document requested by the client, the method comprising the steps of:

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document in order to perform at least one function selected from the following functions:

matching decompression requirements at the client; converting the document into a format compatible for the client;

reducing latency experienced by the client; and altering the document to fit into smaller memory space; and

transmitting the document to the client.

2. A method according to claim 1, wherein the request includes a URL (Uniform Resource Locator).

3. A method according to claim 1, wherein the step of transcoding is performed while the step of retrieving the document is performed.

4. A method according to claim 1, wherein the step of transcoding comprises the step of reading at least a portion of the document from a proxy cache located in the proxying server.

5. A method according to claim 1, wherein the step of transmitting the document to the client is performed prior to performing the step of transcoding, at the proxying server, the data in the document.

6. A method according to claim 1, wherein the step of transmitting the document to the client comprises the step of transmitting the transcoded document to the client, wherein the step of transmitting the document to the client is performed after the step of transcoding, at the proxying server, the data in the document.

7. A method according to claim 1, wherein the document includes a link to another document, the link including a retrieval address, and wherein the step of transcoding at the proxying server the data in the document comprises the step of updating the link

8. A method according to claim 7, wherein the other document is an image, and wherein the step of updating the link comprises the step of adding information to the document indicating the size of the image.

9. A method according to claim 8, wherein the other document is inaccessible to the proxying server, and wherein the step of updating the link comprises the step of removing the link.

10. A method according to claim 7, wherein the other document has been relocated from the retrieval address to a redirect address, and wherein the step of updating the link comprises the step of updating the link to correspond to the redirect address.

11. A method according to claim 1, wherein the client includes a television display, wherein the document references an image, and wherein the step of transcoding, at the proxying server, the data in the document is conducted in order to perform at least the function of converting the document into a compatible format for the client, and comprises the step of revising the data such that the image is sized for display on the television display.

18

12. A method according to claim 1, wherein the document references an image having a first image format, and wherein the step of transcoding, at the proxying server, the document is conducted in order to perform at least the function of converting the document into a compatible format for the client, and includes the step of converting the first image format to a second image format.

13. A method according to claim 1, further comprising the steps of:

identifying an image referenced by the document;

determining whether the image has been previously retrieved by the proxying server; and

if the image has been previously retrieved by the proxying server, accessing information stored in the proxying server indicating the size of the image;

wherein the step of transcoding, at the proxying server, the data in the document is conducted in order to perform at least the function of reducing latency experienced by the client, and comprises the step of using the information indicating the size of the image to revise the data of the document to allow the document to be partially displayed by the client before the image is received by the client.

14. A method according to claim 13, wherein the step of transcoding, at the proxying server, the data in the document comprises the following step:

if the image has been previously retrieved by the proxying server, transcoding at the proxying server the data in the document without the image but providing a space for the image to be later inserted; and

wherein the step of transmitting the document to the client comprises the following steps:

transmitting the document to the client without the image but providing the space for the image to be later inserted; and

after transmitting the document to the client without the image, transmitting the image to the client.

15. In a proxying server coupled to a client and to a remote server, the proxying server operating as a proxy on behalf of the client for accessing the remote server, a method of retrieving and transcoding a document requested by the client, the method comprising the steps of:

providing a persistent database at the proxying server, the persistent database including information relating to the document;

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document using the information included in the persistent database in order to perform at least one function selected from the following functions:

matching decompression requirements at the client; converting the document into a format compatible for the client;

reducing latency experienced by the client; and altering the document to fit into smaller memory space; and

transmitting the document to the client.

16. A method according to claim 15, further comprising the step of storing in the persistent database validity information corresponding to the document.

19

17. A method according to claim 16, wherein the step of retrieving the document comprises the following steps:

retrieving the document from the persistent database if the validity information corresponding to the document indicates that the document is valid; and

retrieving the document from the remote server if the validity information corresponding to the document indicates that the document is not valid.

18. A method according to claim 16, wherein the step of storing in the persistent database validity information corresponding to the document comprises the step of observing a rate of change of the document.

19. A method according to claim 18, wherein the step of observing a frequency of change of the document comprises the step of periodically retrieving the document, wherein successive retrievals of the document are separated in time by at least a time interval.

20. A method according to claim 19, wherein the step of periodically retrieving the document comprises the following steps:

retrieving the document at a first time;

storing the document retrieved at the first time in a memory;

retrieving the document at a second time, the second time being at least the time interval after the first time;

determining whether the document retrieved at the second time has changed compared to the document retrieved at the first time;

if the document retrieved at the second time is different than the document retrieved at the first time, replacing the document retrieved at the first time with the document retrieved at the second time in the memory; and

if the document retrieved at the second time is the same as the document retrieved at the first time, extending the time interval.

21. A method according to claim 15, further comprising the step of storing in the persistent database performance information relating to performance of the remote server when accessing the document.

22. A method according to claim 21, wherein the performance information comprises a latency value.

23. A method according to claim 15, further comprising the step of storing in the persistent database information for optimizing memory usage by the client.

24. A method according to claim 15, wherein the step of retrieving the document comprises the following steps:

determining whether a redirect address corresponding to the document is stored in the persistent database;

if the redirect address corresponding to the document is stored in the persistent database, accessing the remote server using the redirect address, the remote server corresponding to the redirect address; and

if the redirect address corresponding to the document is not stored in the persistent database, the method further comprises the following steps:

accessing a previous remote server at which the document previously resided;

obtaining the redirect address from the previous remote server;

storing the redirect address in the persistent database; and

20

accessing the remote server using the redirect address.

25. A computer program product for implementing, in a proxying server coupled to a client and to a remote server, a method of retrieving and transcoding a document requested by the client, the computer program product comprising a computer-readable medium carrying computer-executable instructions for causing the proxying server to perform acts included in the method, said acts comprising:

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document in order to perform at least one function selected from the following functions:

matching decompression requirements at the client;

converting the document into a format compatible for the client;

reducing latency experienced by the client; and
altering the document to fit into smaller memory space; and

transmitting the document to the client.

26. A computer program product according to claim 25, wherein the computer-executable instructions comprise a plurality of program code means for transcoding at least a portion of the document, including:

program code means for transcoding at least a portion of the document so as to match decompression requirements at the client;

program code means for transcoding at least a portion of the document so as to convert the document into a format compatible with the client;

program code means for transcoding at least a portion of the document so as to reduce latency experienced by the client; and

program code means for transcoding at least a portion of the document so as to alter the document to fit into smaller memory space.

27. A computer program product according to claim 26, wherein the act of transcoding comprises selecting at least one of said plurality of program code means for transcoding for use in the act of transcoding.

28. A computer program product according to claim 25, wherein the act of transcoding is performed while the act of retrieving the document is performed.

29. A computer program product according to claim 25, wherein the act of transcoding comprises reading at least a portion of the document from a proxy cache located in the proxying server.

30. A computer program product according to claim 25, wherein the act of transmitting the document to the client is performed prior to performing the act of transcoding.

31. A computer program product according to claim 25, wherein the act of transmitting the document to the client comprises the act of transmitting the transcoded document to the client, wherein the act of transmitting the document to the client is performed after the act of transcoding.

* * * * *



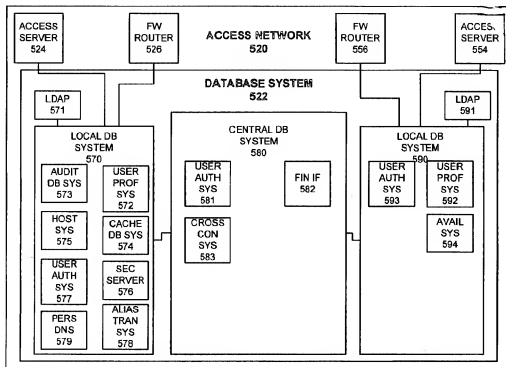
US006697806B1

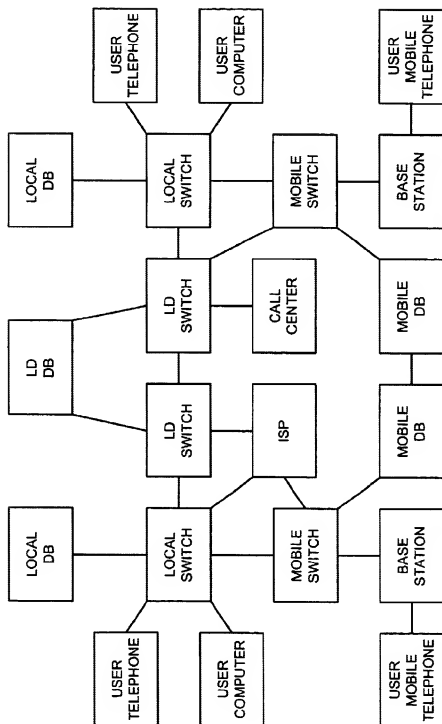
**(12) United States Patent
Cook****(10) Patent No.: US 6,697,806 B1
(45) Date of Patent: Feb. 24, 2004****(54) ACCESS NETWORK AUTHORIZATION****(75) Inventor: Fred S. Cook, Olathe, KS (US)****(73) Assignee: Sprint Communications Company, L.P., Overland Park, KS (US)****(*) Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.**(21) Appl. No.: 09/574,978****(22) Filed: May 19, 2000****Related U.S. Application Data****(63)** Continuation of application No. 09/556,276, filed on Apr. 24, 2000.**(51) Int. Cl.⁷ G06F 17/30****(52) U.S. Cl. 707/10; 707/3; 707/9;
709/227; 709/203; 709/219****(58) Field of Search 707/1-3, 9-10,
707/4, 104.1; 709/223-227, 219, 203****(56) References Cited****U.S. PATENT DOCUMENTS**5,884,312 A * 3/1999 Dustan et al. 707/10
6,151,601 A * 11/2000 Papierniak et al. 707/1

* cited by examiner

*Primary Examiner—Jean R. Homere
Assistant Examiner—Mohammad Ali***(57) ABSTRACT**

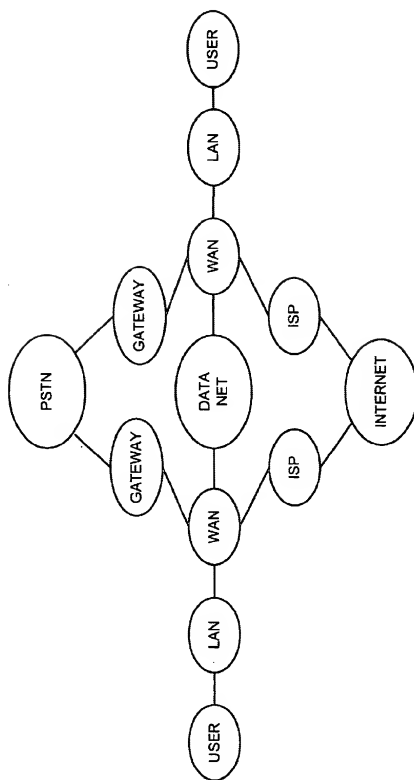
An access communication system provides access between a user system and a plurality of communication networks. The plurality of communication networks provide services to a user in the user system. An access communication system includes a local database system and an access server that is connected to the user system and the plurality of communication networks. The local database system receives a user logon. The local database system then processes the user logon to determine if the user is allowed access to the access communication system based on a local database system. The local database system then provides access to the access communication system to the user in response to the determination that the user is allowed access based on the local database system. The local database system then generates an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system. The local database system receives and processes an authorization response indicating whether the user is allowed to use the access system from the second database system. The local database system then provides access to the access communication system to the user in response to the authorization response that allows the user to use the access communication system.

24 Claims, 63 Drawing Sheets



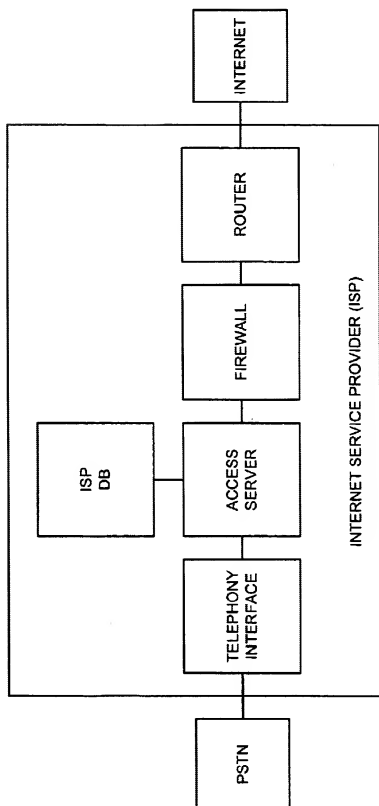
PRIOR ART

FIG. 1



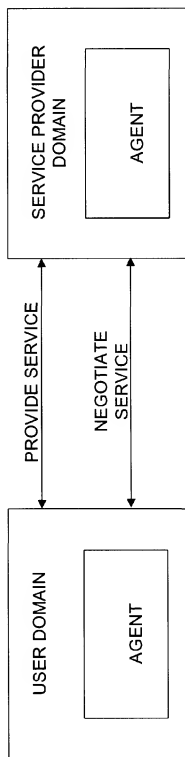
PRIOR ART

FIG. 2



PRIOR ART

FIG. 3



PRIOR ART

FIG. 4

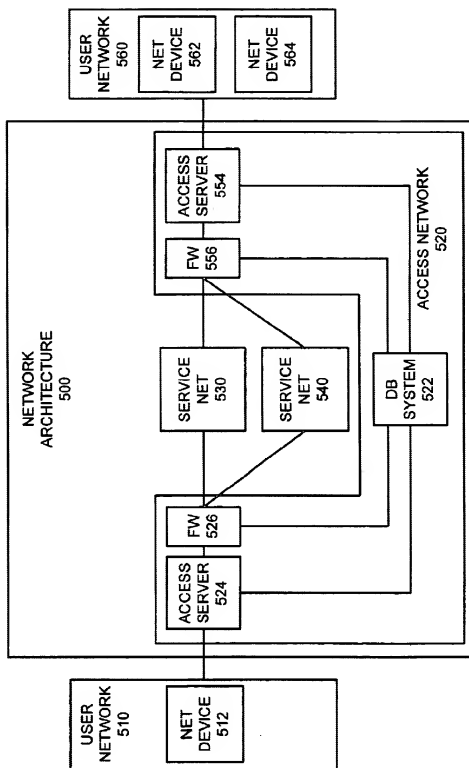


FIG. 5

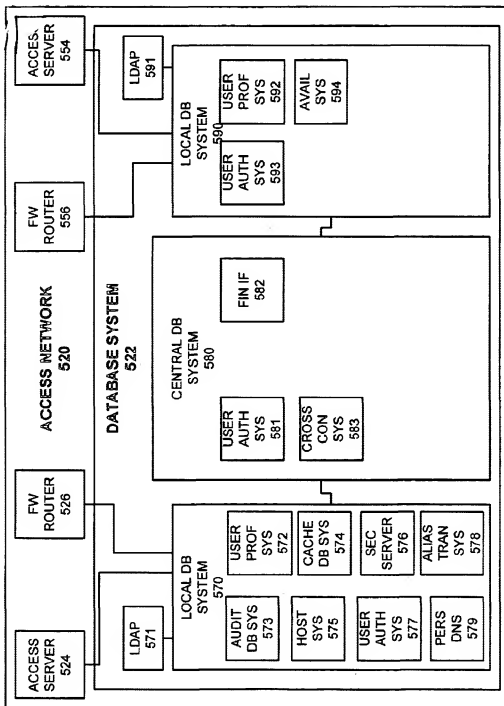


FIG. 6

USER ID	PASSWORD	NAME	ACCOUNT NUMBER	SERVICES	ADDRESS	BILLING CODE	CLASS	GROUP	SHELL	MACROS

FIG. 7

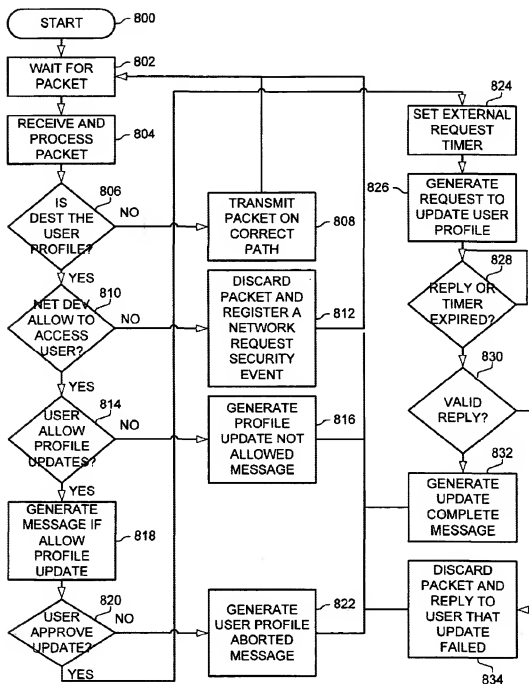


FIG. 8

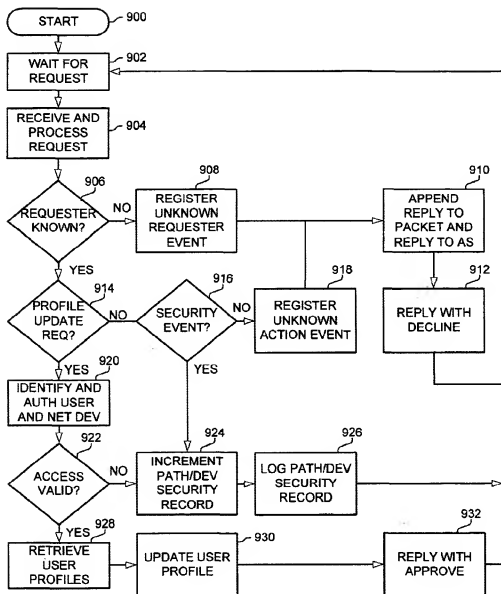


FIG. 9

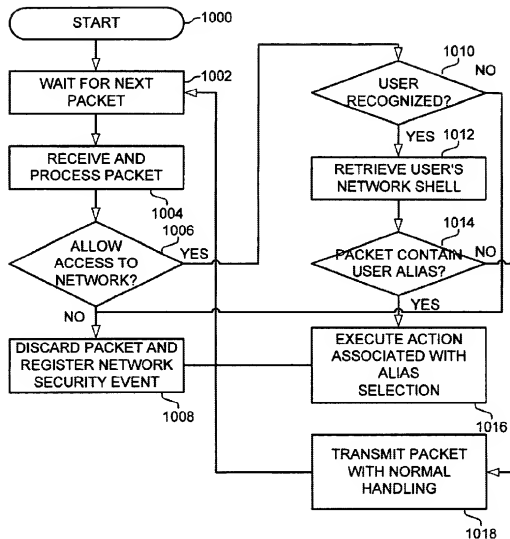


FIG. 10

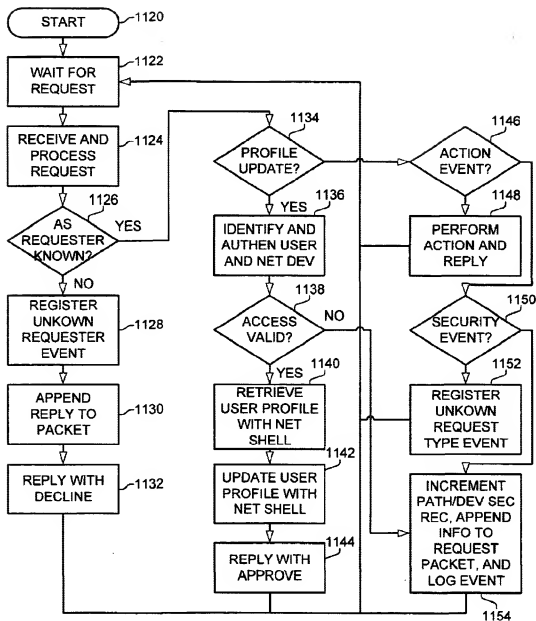


FIG. 11

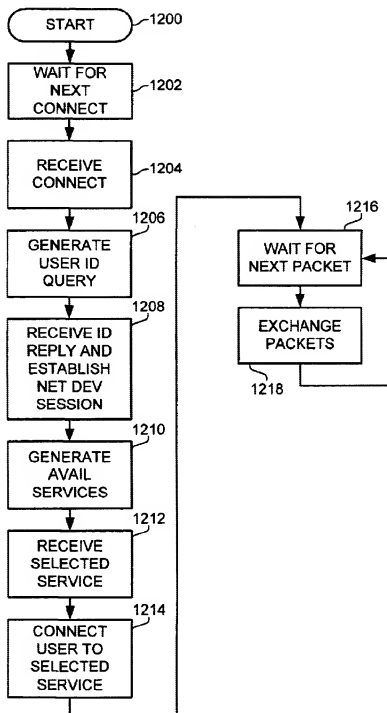


FIG. 12

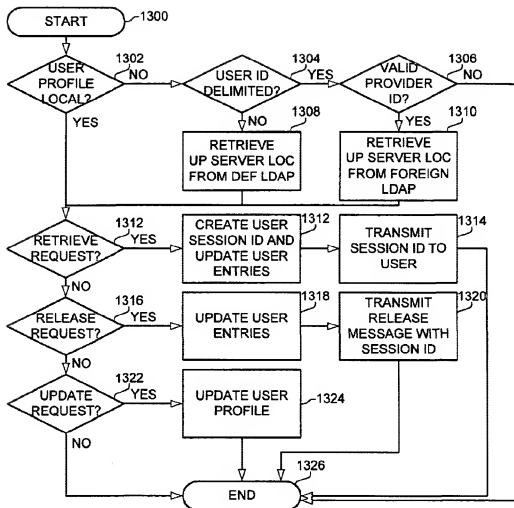


FIG. 13

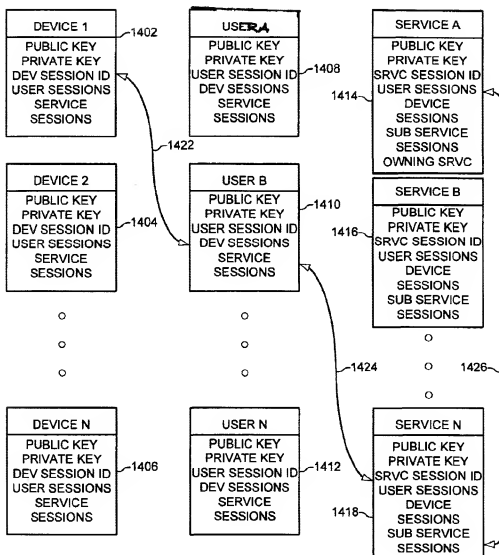
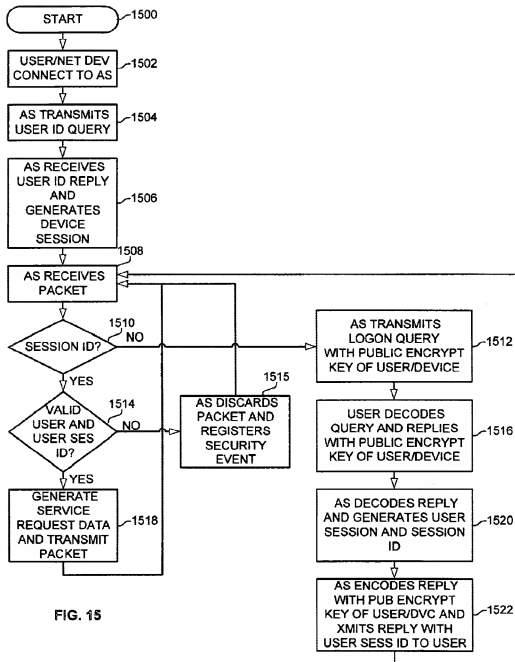


FIG. 14



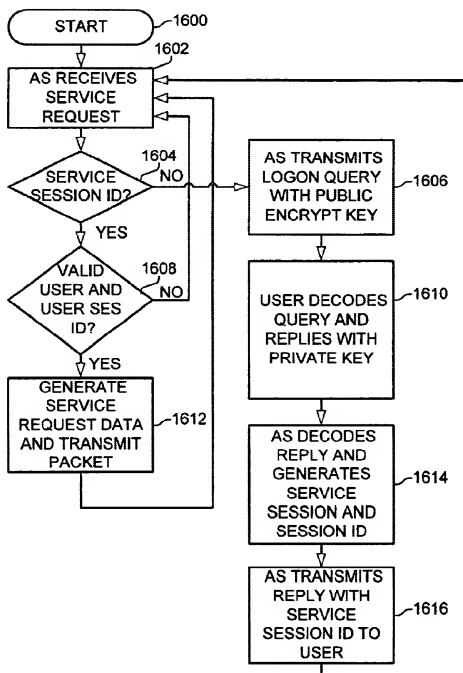


FIG. 16

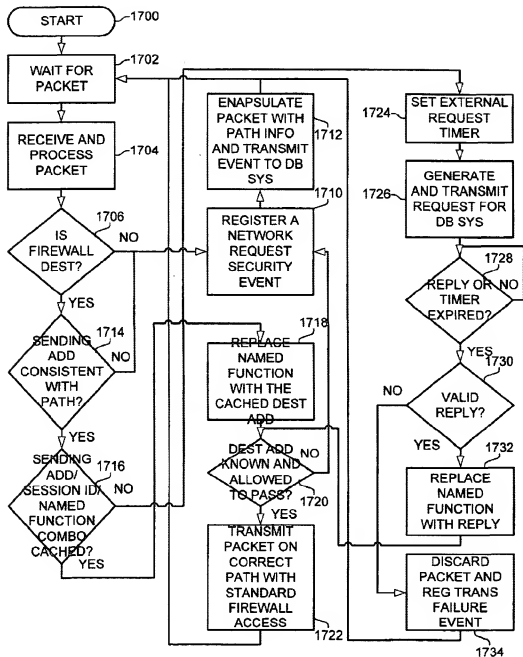


FIG. 17

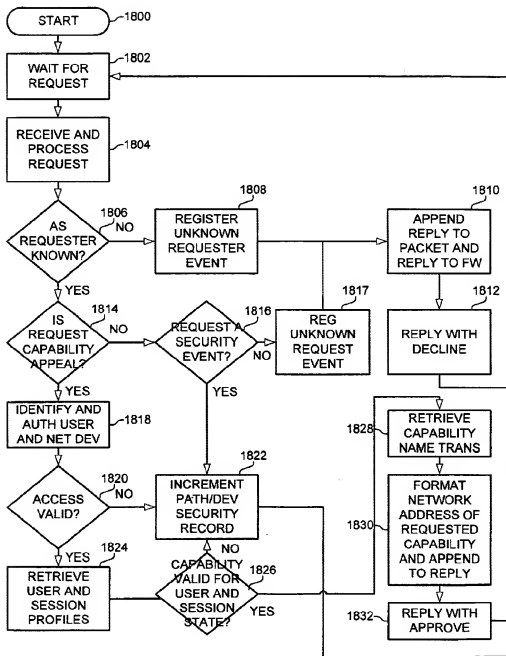


FIG. 18

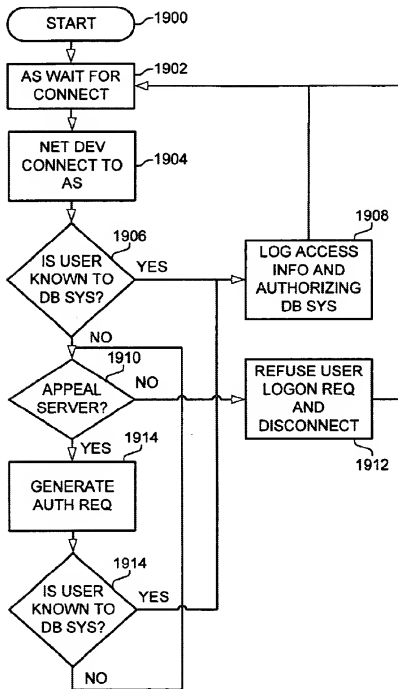


FIG. 19

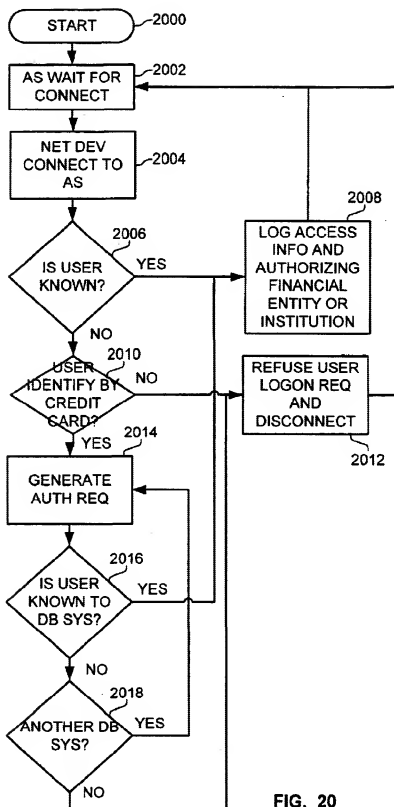


FIG. 20

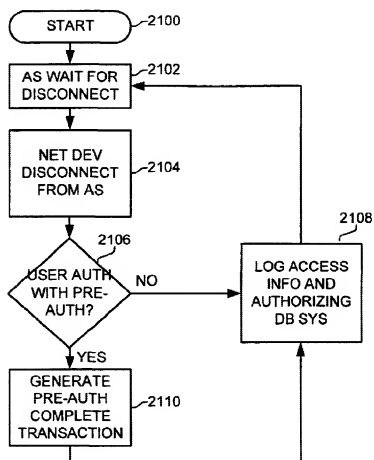


FIG. 21

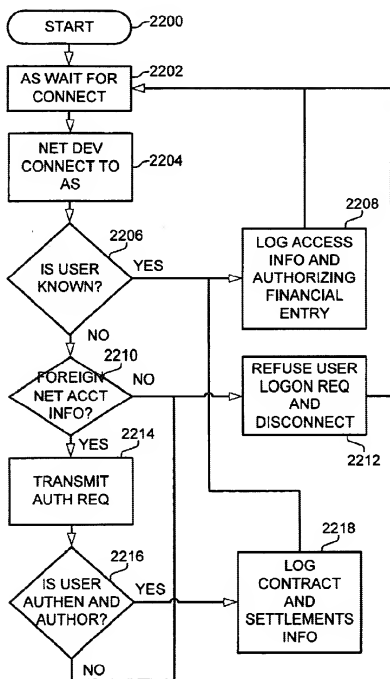
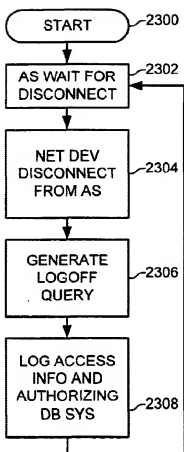


FIG. 22

**FIG. 23**

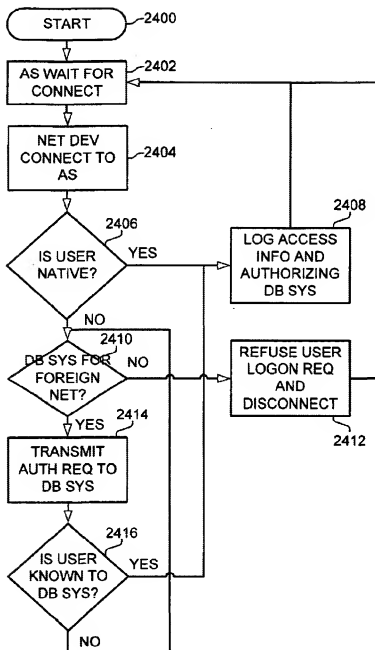


FIG. 24

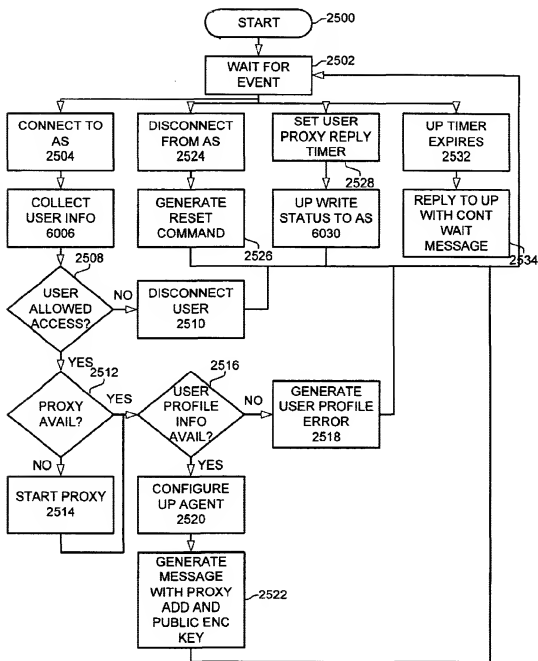


FIG. 25

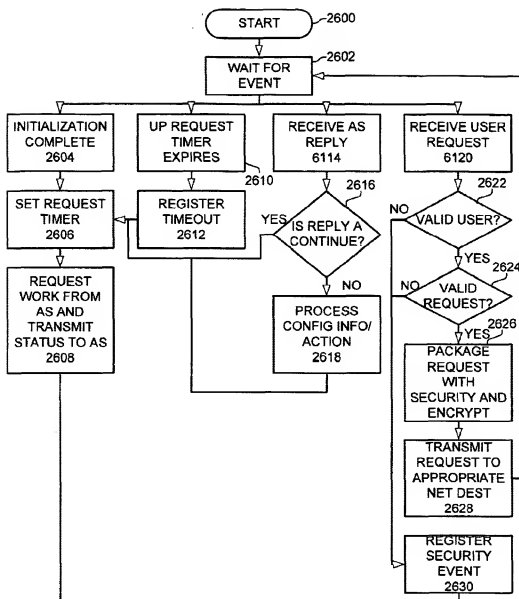


FIG. 26

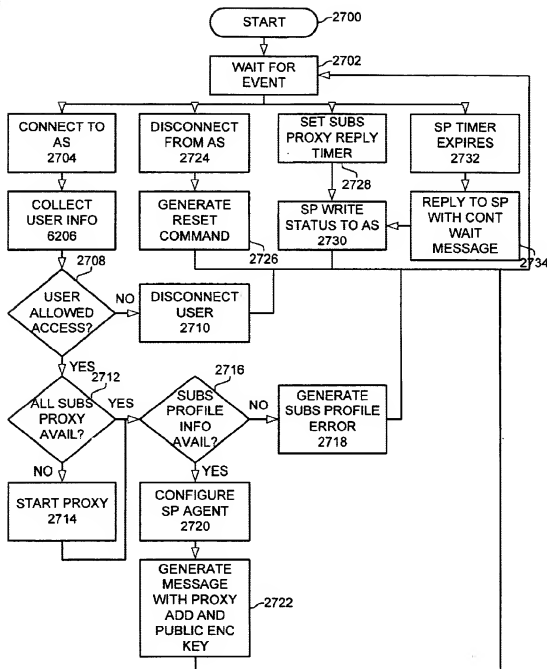


FIG. 27

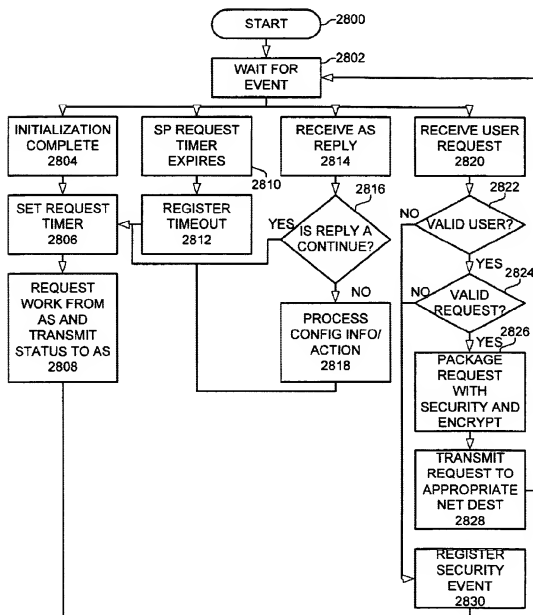


FIG. 28

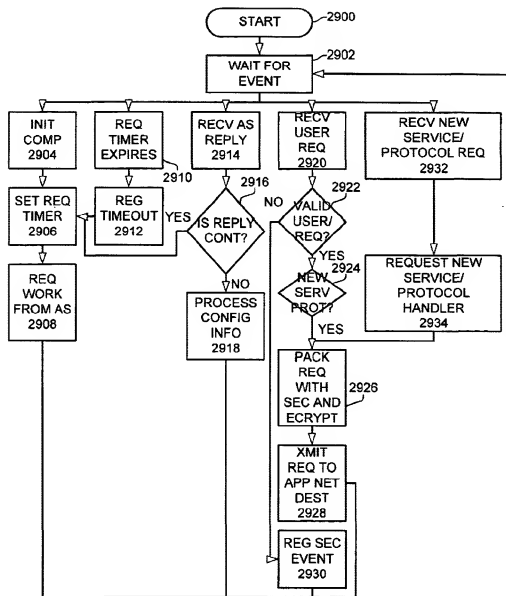


FIG. 29

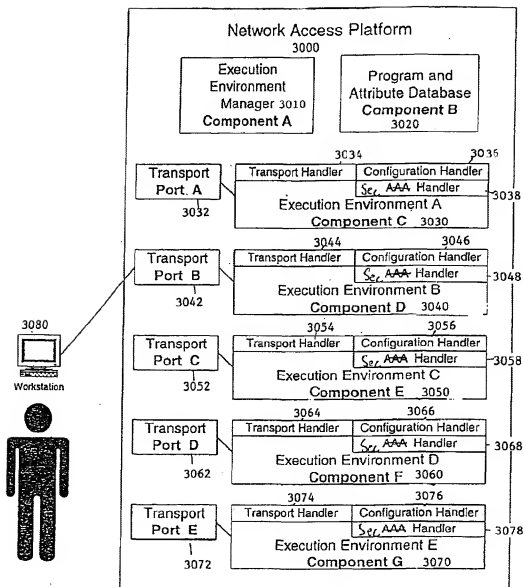


FIGURE 30

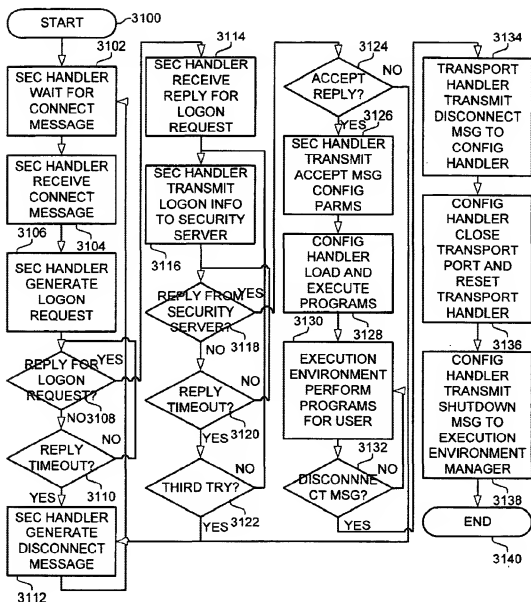


FIG. 31

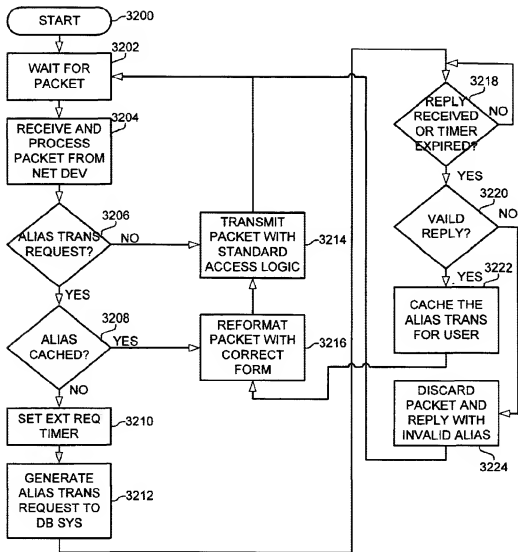


FIG. 32

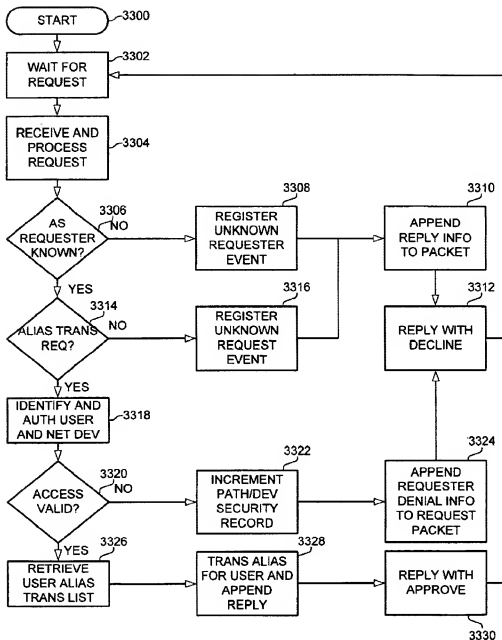


FIG. 33

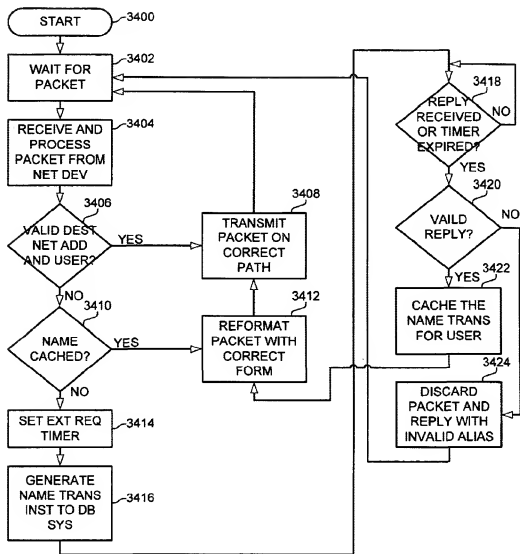


FIG. 34

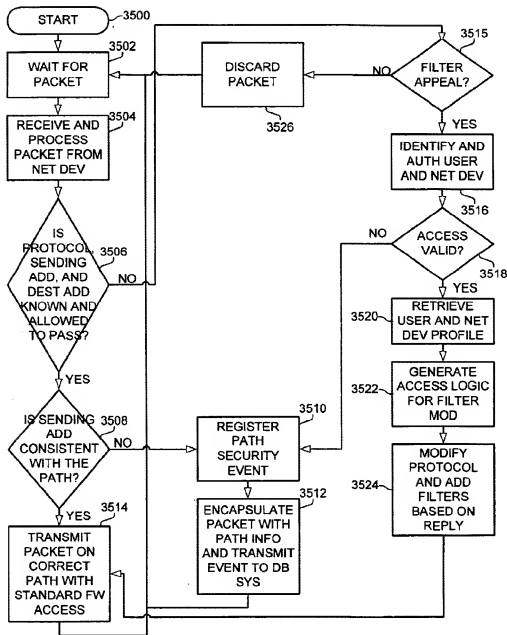


FIG. 35

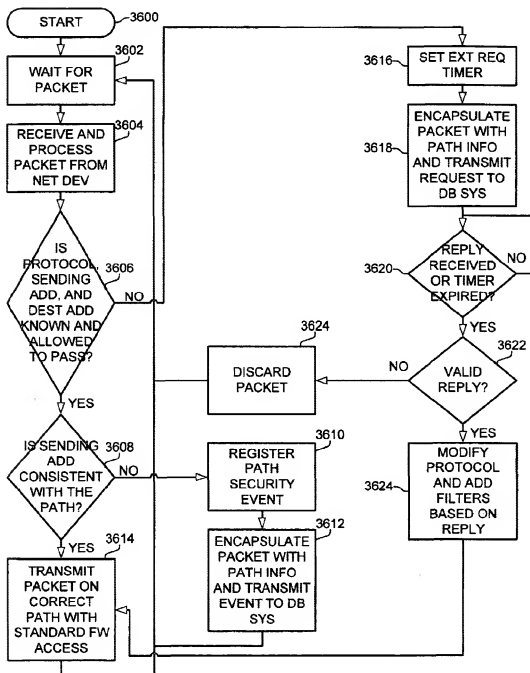


FIG. 36

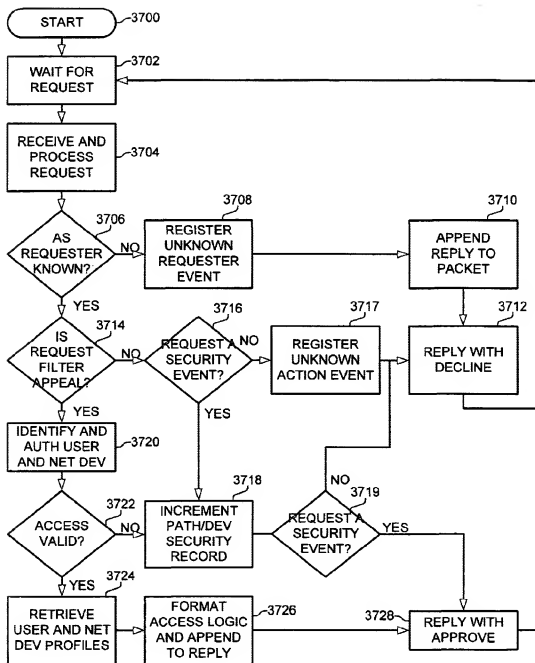


FIG. 37

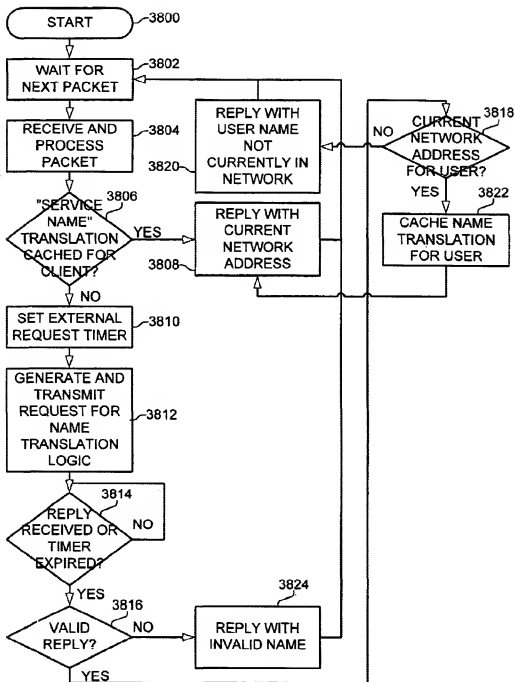


FIG. 38

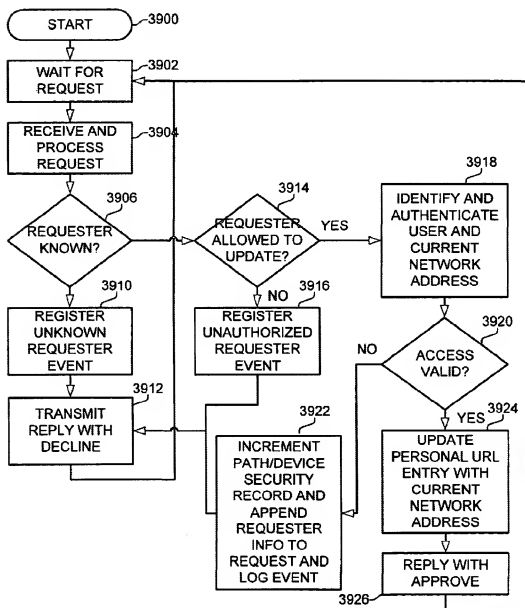


FIG. 39

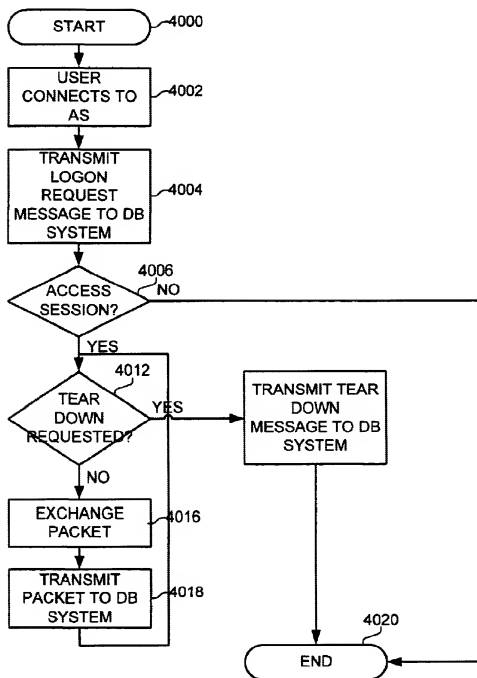


FIG. 40

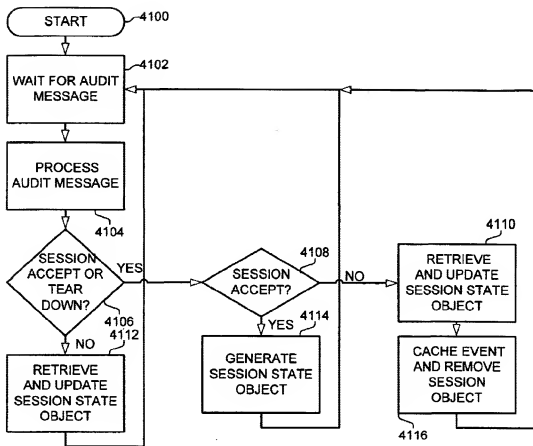


FIG. 41

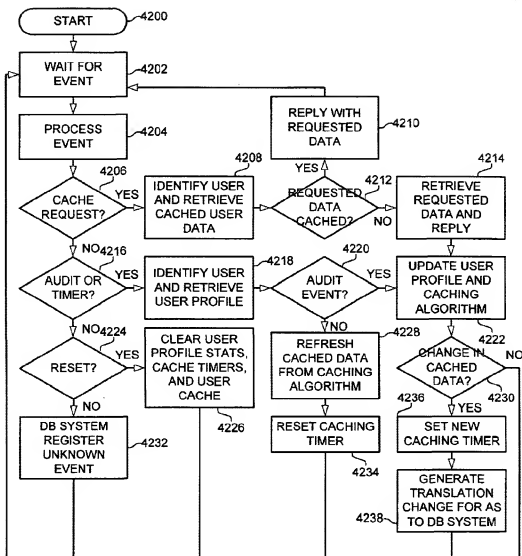


FIG. 42

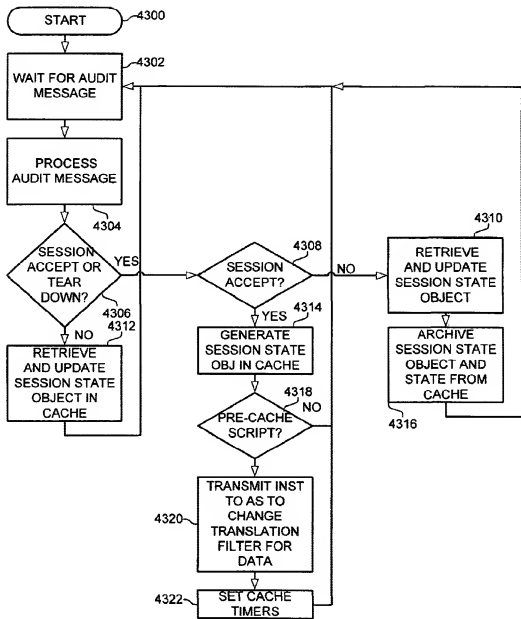


FIG. 43

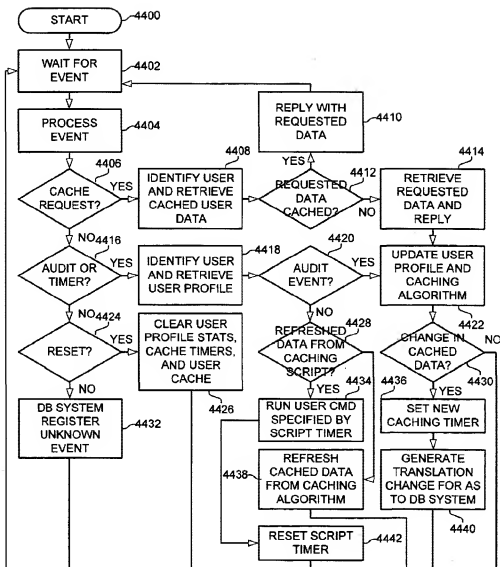


FIG. 44

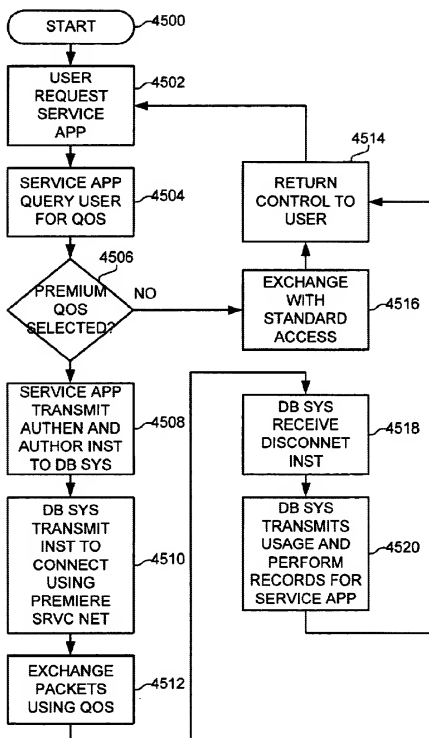


FIG. 45

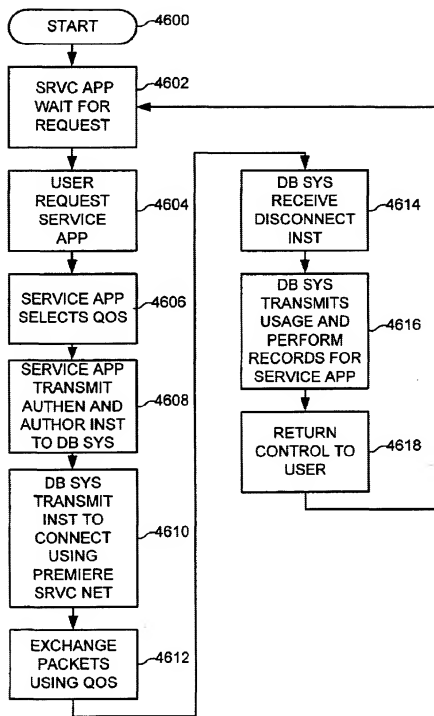


FIG. 46

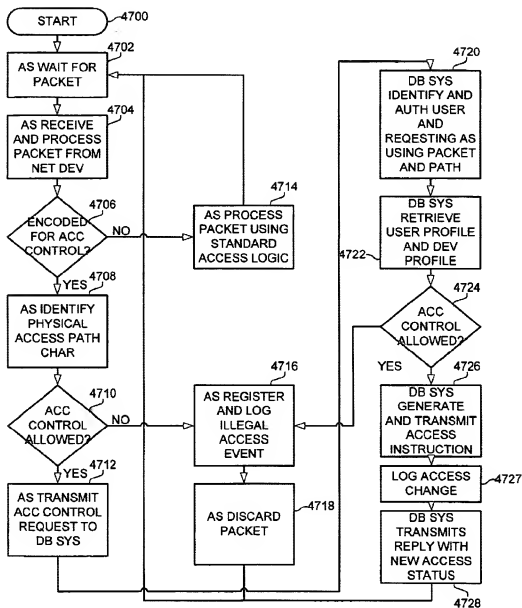


FIG. 47

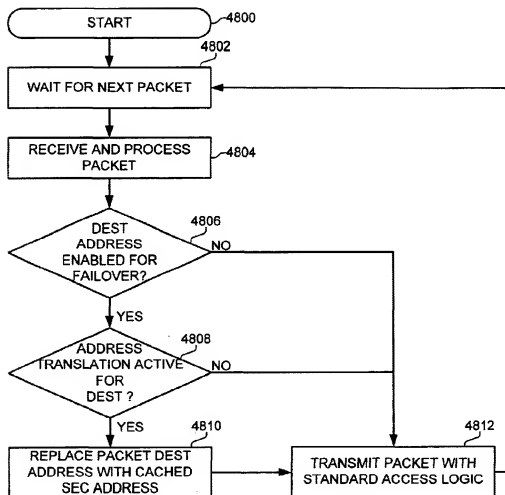


FIG. 48

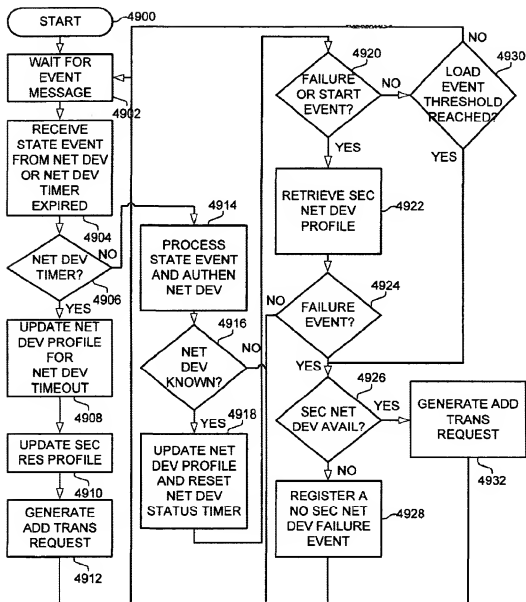


FIG. 49

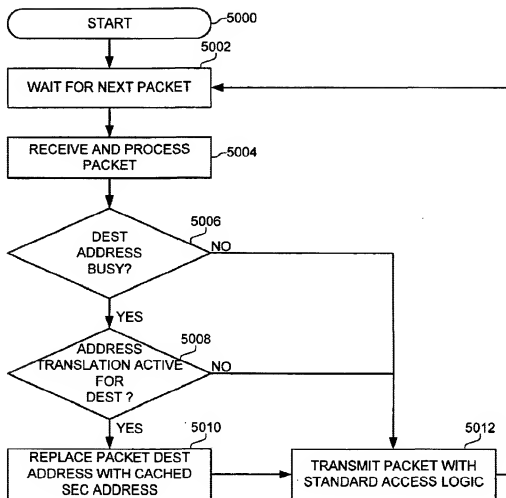


FIG. 50

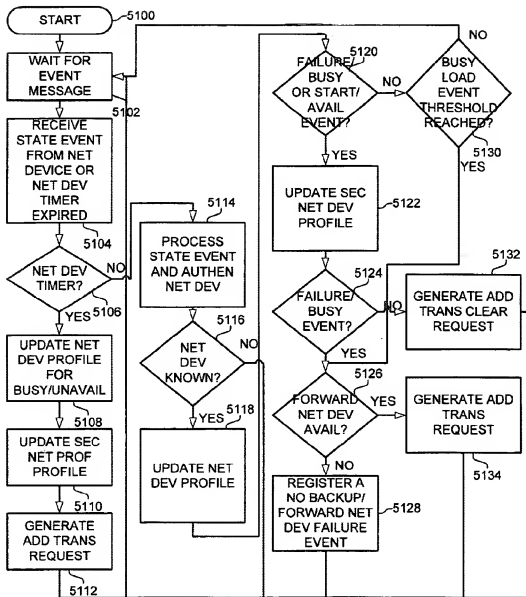


FIG. 51

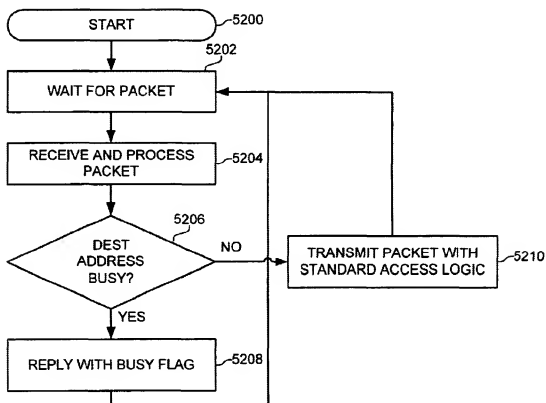


FIG. 52

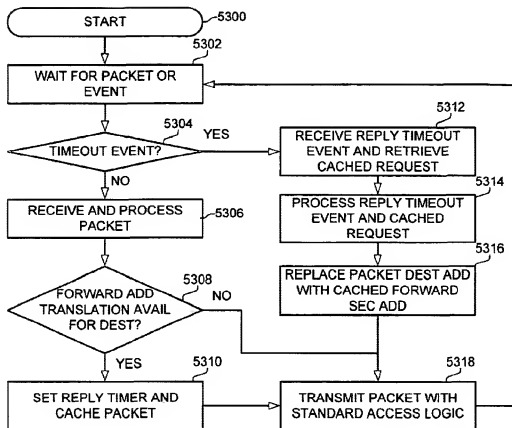


FIG. 53

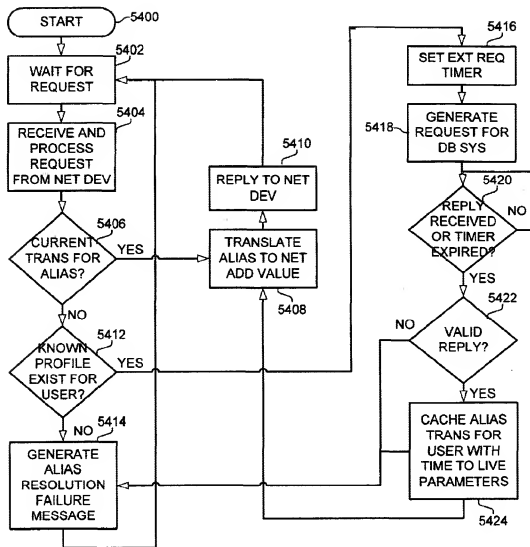


FIG. 54

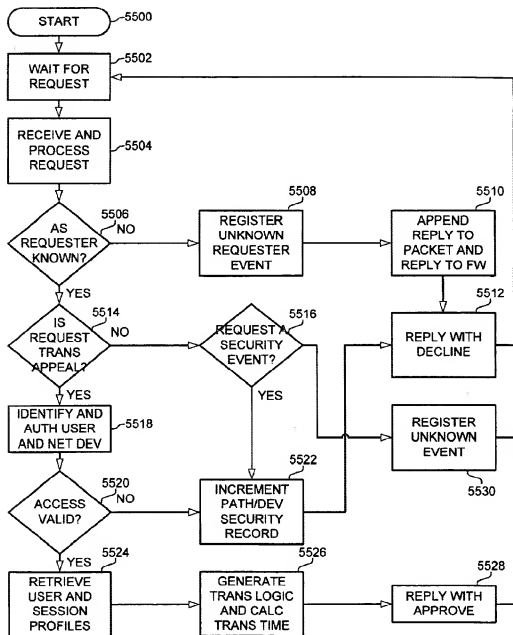


FIG. 55

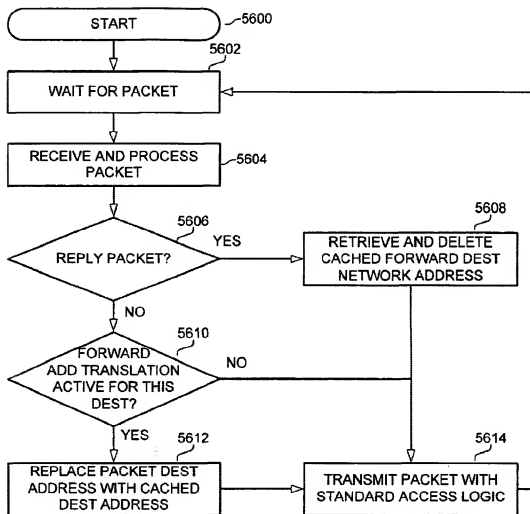


FIG. 56

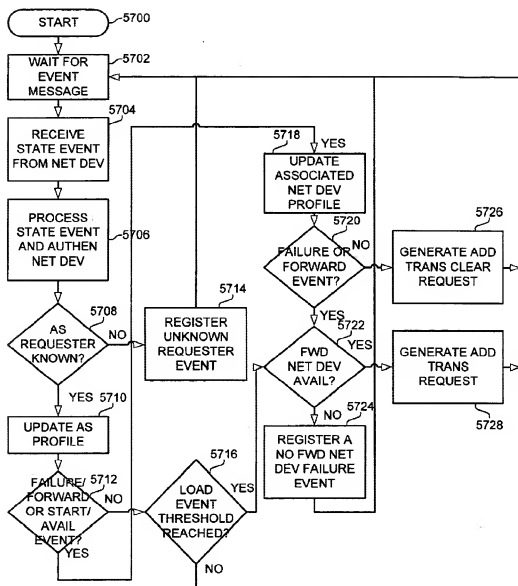


FIG. 57

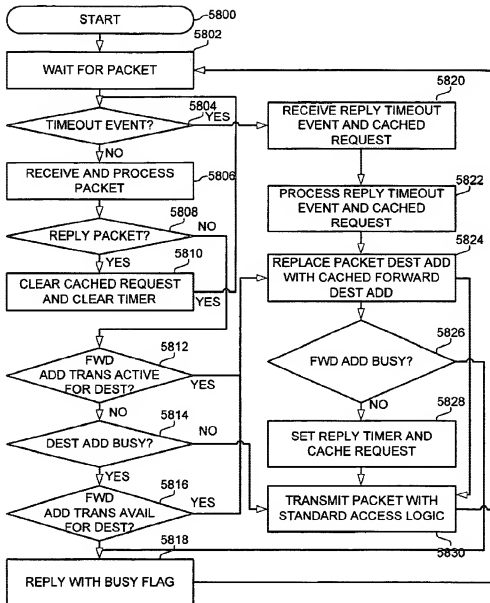


FIG. 58

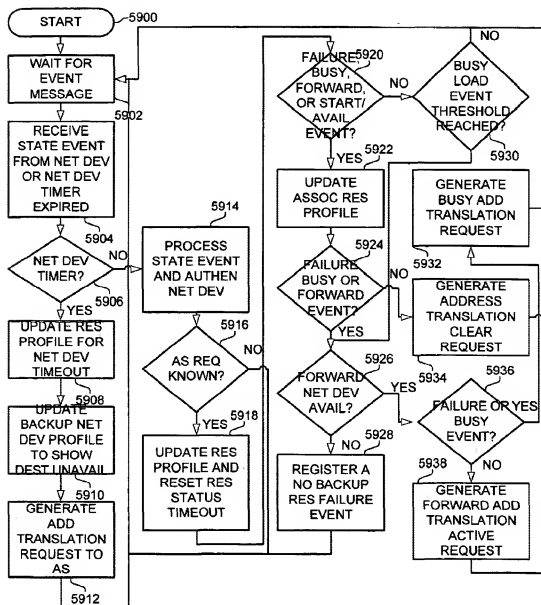
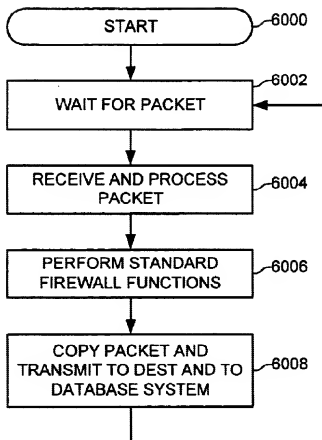


FIG. 59

**FIG. 60**

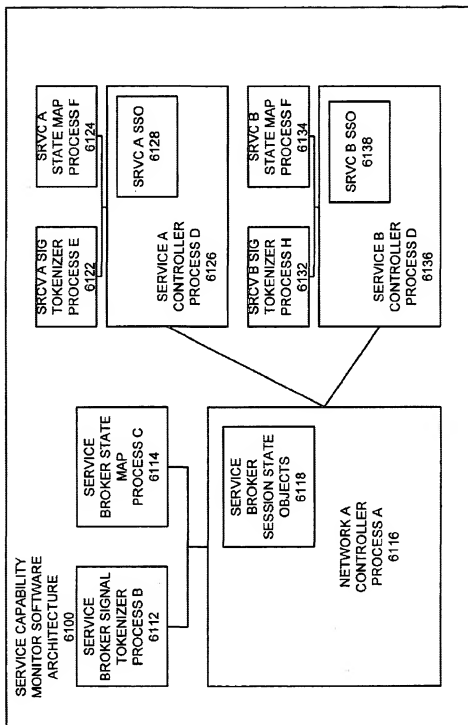
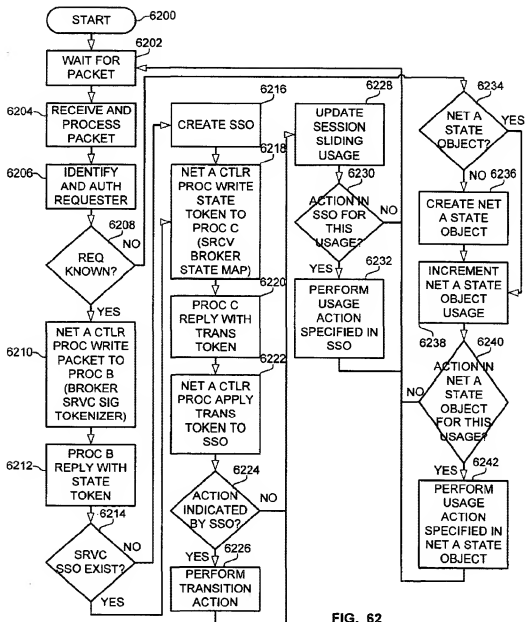


FIG. 61



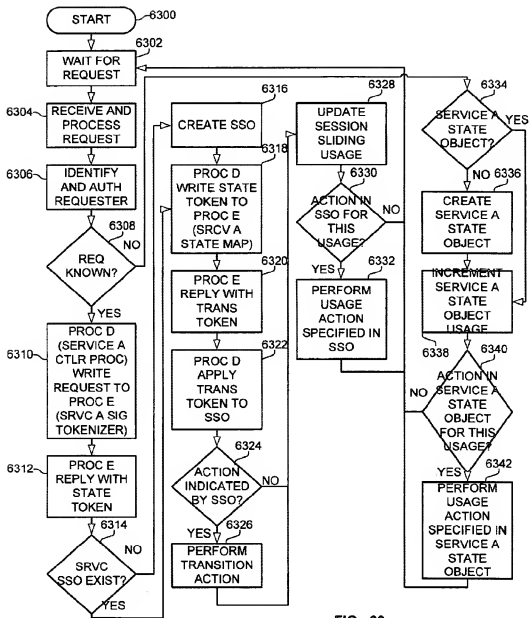


FIG. 63

ACCESS NETWORK AUTHORIZATION

RELATED APPLICATIONS

This application is a continuation of prior application Ser. No. 09/556,276, entitled "Access Communication System", filed Apr. 24, 2000, currently pending, and incorporated by reference into this application.

FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

Not applicable

MICROFICHE APPENDIX

Not applicable

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention is related to the field of communication networks, and in particular, to an access communication system that provides access to multiple service provider systems. More particularly, this invention relates to a system that authorizes access to use the access communication system.

2. Description of the Prior Art

Communication networks have seen dramatic development over the past several years so that today, there are multiple diverse communication networks providing services. The current technical challenge is to develop interfaces between the networks to provide seamless service across multiple networks. Unfortunately, today's interfaces lack the ability to offer the user with easy access to services from multiple systems. These interfaces do not customize operations for the user.

FIG. 1 illustrates the conventional public telephone network. User telephones and computers are connected to local switches. The local switches are coupled to a local database. The user places calls through the local switch. The local switch processes the called number to provide an end-point connection or access to other networks. The local switch may connect the call to another telephone in the local calling area. In a Local Number Portability (LNP) situation, the local switch exchanges information with the local database to obtain the appropriate routing number for a ported call. The local switch may also connect the call to an Internet Service Provider. If a Digital Subscriber Line (DSL) is used, DSL equipment may be used to bypass the local switch. The local switch may also connect the call to a long distance switch. To provide access to the long distance switch, the local switch first exchanges information with the local database. The local database identifies the long distance network for either the user or the dialed number.

The long distance switch processes the called number to route the call to another system or network. Prior to routing, the long distance switch validates the call by checking the caller's number. The long distance switch may also exchange information with the long distance database to provide special call-handling. One example is a calling card call, where the long distance database validates an account number for billing the call. Another example is a toll-free call, where the long distance database processes external information customized by the called party to route the call. Toll-free routing information includes items such as time and date, caller location, and call center status. Many long distance calls are simply routed through the long distance

network to another local network for call completion. Other calls are routed from the long distance network to a call center. Call centers offer a concentration of call-handling capabilities for operations, such as order entry, customer service, and promotions. Call centers include automatic call distribution equipment to route calls to the appropriate destination within the call center.

Both local and long distance networks exchange calls with mobile switches. The mobile switches are connected to base stations that communicate over the air with wireless telephones. When a mobile user places a call, the mobile switches exchange information with a mobile database to validate the mobile caller. The call is then routed to another mobile caller, the telephone network, or to an ISP. When a mobile caller moves around, their wireless phone logs-in with the physically proximate mobile network, and that mobile network updates the mobile user's location in the mobile databases. When a call is placed to that mobile user, their home mobile switch obtains a routing number from the mobile databases to route the call to the mobile network currently in communication with the mobile user.

FIG. 2 illustrates a conventional data network that transfers packets of user data to a destination based on address information carried in the packets. Users are connected to Local Area Networks (LANs) that are connected to Wide Area Networks (WANs). A common LAN is an Ethernet system. A common WAN is an intranet. WANs are interconnected by data networks, such as IP, TI, frame relay, or Asynchronous Transfer Mode (ATM). WANs are connected to the Internet through ISPs. WANs are connected to the public telephone network through telephony gateways. A common telephony gateway is a Private Branch Exchange (PBX).

FIG. 3 illustrates a conventional ISP. The public telephone network is coupled to a telephony interface that converts between telephony analog and digital protocols and the Internet Protocol (IP). Some telephony interfaces also handle DSL traffic that may already use IP. The telephony interface transfers IP traffic through an access server and firewall to a router. Some ISPs combine the firewall and the access server into one system. Also, the position of the firewall may vary, and traffic shapers may be present. The router exchanges IP traffic with the Internet.

In operation, the user calls the ISP over the telephone network and logs-in at the access server. The access server collects and forwards the user name and password to the ISP database. The ISP database validates the user name and password and returns an IP address to the access server. The IP address is for the user's terminal connection. Using the IP address, the user may communicate through the firewall to the router for transmissions to an IP address. The user now has Internet access through the router and exchanges packets with various Internet servers.

IP addresses are referred to as network addresses and include a network ID and a host ID. Network IDs are unique across the Internet and host IDs are unique within a given network. IP addresses are lengthy numerical codes, so to simplify things for the user, service addresses are available that are easier to remember. The service addresses are often the name of the business followed by ".com". Domain Name Service (DNS) is hosted by servers on the Internet and translate between service addresses and network addresses. The browser in the user computer accesses the DNS to obtain the desired network address.

FIG. 4 illustrates conventional network access. A current proposal for communication network access is provided by

3

the Telecommunication Information Network Architecture Consortium (TINA-C). TINA-C proposes the use of agents in the user domain and the service provider domain. The service provider domain could be a telephone network, data network, or ISP. The agents negotiate access service rights. Once the service is negotiated, the user receives the service from the service provider network during a service session. Unfortunately, the access session occurs between the user domain and a particular service provider domain. At present, the service provider domain provides limited access capability beyond simply handing off communications to another network based on a called number or network address. As a result, the ability to customize services for a particular user across multiple service providers is inadequate.

SUMMARY OF THE INVENTION

The inventions solve the above problems by providing access between a user system and a plurality of communication networks. The plurality of communication networks provide services to a user in the user system. An access communication system includes a database system and an access server that is connected to the user system and the plurality of communication networks.

In one aspect of the inventions for user access profile inheritance, the database system receives an update request from the access server to update a user access profile through inheritance. The database system then processes the update request to inherit user profile information from a user profile data structure. The database system updates the user access profile with the user profile information.

In another aspect of the inventions for network shells, the access server receives an alias selection from a user for a network shell that includes alias selections associated with actions. The access server then processes the alias selection to execute an action associated with the alias selection.

In another aspect of the inventions for service based directory, the access server transmits a list of services to a user system. The access server then receives a selection from the list of services. The access server processes the selection to generate an instruction to provide the service related to the selection.

In another aspect of the inventions for user access profile mobility, the database system receives user information. The database system then processes the user information to determine if a user access profile is local within a local database system. The database system generates and transmits a request to retrieve a user access profile from a second database system external to the local database system in response to the determination that the user access profile is not local.

In another aspect of the inventions for service, user, and device sessions, the access communication system establishes a connection between a network device and the access server. The access communication system then generates a device session including a device session ID based on the network device. The access communication system generates and transmits a login query for the network device. The access communication system receives and processes a login reply from the network device to generate a user session including a user session ID based on the user. The access communication system receives and processes a request for the service to generate a service session including a service session ID based on the service. The service may generate and transmit a login query for the user. The access communication system links the device session, user session, and the service session using the device session ID, the user session ID, and the service session ID.

4

In another aspect of the inventions for service capability firewall, the access server receives information including a named function request for a service provider. The access server processes the information to check if the named function request is valid for the service provider and the service. If valid, the access server determines if a private destination address exists for the named function request. The access server replaces the named function request with the private destination address in response to the determination that the private destination address exists for the named function request. The access server then transmits the information with the private destination address to the service provider.

In another aspect of the inventions for prepaid access and bank card access, the database system receives information identifying a billing code for a user. The database system then processes the billing code to determine if the user is allowed to use the access system. The database system provides access to the access system in response to the determination that the user is allowed to use the access system.

In another aspect of the inventions for global authentication and access card, the database system receives a user login. The database system then processes the user login to determine if the user is allowed access to the access communication system based on a local database system. The database system then provides access to the access communication system to the user in response to the determination that the user is allowed access based on the local database system. The database system then generates an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system. The database system receives and processes an authorization response indicating whether the user is allowed to use the access system from the second database system. The database system then provides access to the access communication system to the user in response to the authorization response that allows the user to use the access communication system.

In another aspect of the inventions for user based proxy and subscriber based proxy, the database system includes a user proxy. The user proxy receives a request for the service from the user system. The user proxy then transmits the request for the service to a service provider. The user proxy exchanges user information between the user system and the service provider.

In another aspect of the inventions for dynamic proxy, the database system includes a proxy. The proxy receives a service/protocol request for a new service or protocol. The proxy processes the service/protocol request to generate a handler request to obtain a handler for the new service or protocol. The proxy then receives and executes the handler for the new service or protocol.

In another aspect of the inventions for access execution environment, the database system receives and processes a login reply into an access execution environment for a user. The database system retrieves programs for the user into the access execution environment. The database system executes the programs for the user in the access execution environment.

In another aspect of the inventions for domain name scoping and inband domain name service lookup, the access server receives information including an alias from the user system. The access server determines if the alias exists in a cache including aliases and alias translations for the user.

5

The access server changes the information based on the cached alias translation.

In another aspect of the inventions for inline access service triggering, the access server receives information. The access server then processes the information to determine if the information is allowed to pass. The access server changes access logic based on the information in response to the determination that the information is not allowed to pass. The access server changes the filters of the access server based on the information in response to the determination that the information is not allowed to pass.

In another aspect of the inventions for access service triggering, the access server receives information. The access server processes the information to determine if the information is allowed to pass. The access server then generates a request from a database system in response to the determination that the information is not allowed to pass. The access server receives a reply including access logic from the database system. The access server changes filters of the access server based on the access logic.

In another aspect of the inventions for personal URL, the database system receives information including a user alias. The database system processes the information to determine if a user alias translation including a current network address for the user alias exists. The database system then modifies the information with the current network address using the user alias translation.

In another aspect of the inventions for predictive caching, the access server receives a request for data. The access server then determines if the data exists in a user cache wherein the user cache contains cached data based on the user's predictive patterns. The access server retrieves the data from the user cache in response to the determination that the data exists in the user cache. The access server transmits the request for data for the service provider in response to the determination that the data does not exist in the user cache.

In another aspect of the inventions for user controlled caching, the access server receives a request for data. The access server determines if the data exists in a user cache wherein the user cache contains cached data based on a user's script of commands. The access server retrieves the data from the user cache in response to the determination that the data exists in the user cache. The access server then transmits the request for data for the service provider in response to the determination that the data does not exist in the user cache.

In another aspect of the inventions for service usage audit, the access server receives an audit message into a database system. The access server processes the audit message to store the audit message in the database system.

In another aspect of the inventions for switching access by a user, switching access by a service provider, and dynamic access control, the database system receives a request. The database system processes the request to determine if the switching of the access is allowed. The database system then generates an instruction to switch access in response to the determination that the switching is allowed.

In another aspect of the inventions for network failover, network busy forwarding, time-out, busy flag, forwarding, and network endpoint availability management, the access server receives information for a destination network device. The access server determines if the destination network device is available. The access server performs an action in response to the determination that the destination network device is unavailable.

6

In another aspect of the inventions for scheduled alias translation, the access server receives information including an alias. The access server processes the information to determine whether an alias translation exists based on an alias translation schedule. The access server then modifies the information based on the alias translation in response to the determination the alias translation exists.

In another aspect of the inventions for service capability monitor, the database system receives information from an access server during a service session. The database system determines a current state of the service session based on the information. The database system determines a state transition based on the current state and a map of state transitions of the service. The database system determines whether the state transition is valid for the service session.

BRIEF DESCRIPTION OF THE DRAWINGS

The same reference number represents the same element on all drawings.

FIG. 1 illustrates a public telephone network in the prior art.

FIG. 2 illustrates a data network in the prior art.

FIG. 3 illustrates an Internet Service Provider in the prior art.

FIG. 4 illustrates conventional network access.

FIG. 5 illustrates a network architecture in an example of the invention.

FIG. 6 illustrates an access network in an example of the invention.

FIG. 7 illustrates a table for a user access profile in an example of the invention.

FIG. 8 illustrates a flowchart for an access server for inheriting a user access profile in an example of the invention.

FIG. 9 illustrates a flowchart for a database system for inheriting a user access profile in an example of the invention.

FIG. 10 illustrates a flowchart of an access server for executing a network shell in an example of the invention.

FIG. 11 illustrates a flowchart of a database system for updating a network shell in an example of the invention.

FIG. 12 illustrates a flowchart for the services based directory in an example of the invention.

FIG. 13 illustrates a flowchart of user access profile mobility in an example of the invention.

FIG. 14 illustrates a logical view of device, user, and service sessions in an example of the invention.

FIG. 15 illustrates a flowchart for a user based session in an example of the invention.

FIG. 16 illustrates a flowchart for a service based session in an example of the invention.

FIG. 17 illustrates a flowchart for a firewall/router for service capability firewall in an example of the invention.

FIG. 18 illustrates a flowchart for a database system for service capability firewall in an example of the invention.

FIG. 19 illustrates a flowchart for prepaid access in an example of the invention.

FIG. 20 illustrates a flowchart for bank card access for a connection in an example of the invention.

FIG. 21 illustrates a flowchart for network access cards for a disconnection in an example of the invention.

FIG. 22 illustrates a flowchart for network access cards for a connection in an example of the invention.

7

FIG. 23 illustrates a flow chart for network access cards for a disconnection in an example of the invention.

FIG. 24 illustrates a flowchart for global access in an example of the invention.

FIG. 25 illustrates a flowchart for an access server for user based proxies in an example of the invention.

FIG. 26 illustrates a flowchart for a user proxy for user based proxies in an example of the invention.

FIG. 27 illustrates a flowchart for an access server for subscriber based proxies in an example of the invention.

FIG. 28 illustrates a flowchart for a subscriber proxy for subscriber based proxies in an example of the invention.

FIG. 29 illustrates a flowchart for a dynamic proxy for dynamic proxies in an example of the invention.

FIG. 30 illustrates a block diagram of the access execution environment in an example of the invention.

FIG. 31 illustrates a flow chart for the access execution environment in an example of the invention.

FIG. 32 illustrates a flowchart for an access server for domain name scoping in an example of the invention.

FIG. 33 illustrates a flowchart for a database system for domain name scoping in an example of the invention.

FIG. 34 illustrates a flowchart for an access server for an inband domain name service lookup in an example of the invention.

FIG. 35 illustrates a flowchart for inline access service triggering in an example of the invention.

FIG. 36 illustrates a flowchart for an access server for access service triggering in an example of the invention.

FIG. 37 illustrates a flowchart for a database system for access service triggering in an example of the invention.

FIG. 38 illustrates a flow chart for the personal URL lookup in an example of the invention.

FIG. 39 illustrates a flow chart for the personal URL update in an example of the invention.

FIG. 40 illustrates a flowchart for an access server for auditing in an example of the invention.

FIG. 41 illustrates a flowchart for a database system for auditing in user predictive caching in an example of the invention.

FIG. 42 illustrates a flowchart for a database system for caching in user predictive caching in an example of the invention.

FIG. 43 illustrates a flowchart for a database system for auditing in user controlled caching in an example of the invention.

FIG. 44 illustrates a flowchart for a database system for caching in user controlled caching in an example of the invention.

FIG. 45 illustrates a flowchart for switching access by a user in an example of the invention.

FIG. 46 illustrates a flowchart for switching access by a service provider in an example of the invention.

FIG. 47 illustrates a flowchart for dynamic switching access in an example of the invention.

FIG. 48 illustrates a flowchart for an access server for network address failover in an example of the invention.

FIG. 49 illustrates a flowchart for a database system for network address failover in an example of the invention.

FIG. 50 illustrates a flowchart for an access server for network busy forwarding in an example of the invention.

FIG. 51 illustrates a flowchart for a database system for network busy forwarding in an example of the invention.

8

FIG. 52 illustrates a flowchart for an access server for a busy flag when the destination network device is busy in an example of the invention.

FIG. 53 illustrates a flowchart for an access server for forwarding, if the destination network device timeouts in an example of the invention.

FIG. 54 illustrates a flowchart for an access server for schedule alias resolution in an example of the invention.

FIG. 55 illustrates a flowchart for a database system for scheduled alias resolution in an example of the invention.

FIG. 56 illustrates a flowchart for an access server for destination controlled forwarding in an example of the invention.

FIG. 57 illustrates a flowchart for a database system for destination controlled forwarding in an example of the invention.

FIG. 58 illustrates a flowchart for an access server for network endpoint availability management in an example of the invention.

FIG. 59 illustrates a flowchart for a database system for network endpoint availability management in an example of the invention.

FIG. 60 illustrates a flowchart for a firewall/router for service capability monitor in an example of the invention.

FIG. 61 illustrates a service capability monitor software architecture for a service capability monitor in an example of the invention.

FIG. 62 illustrates a flowchart for the network logic for service capability monitor in an example of the invention.

FIG. 63 illustrates a flowchart for the service logic for service capability monitor in an example of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Communication Network Architecture—FIGS. 5 and 6

The following description and associated figures discuss specific examples intended to teach the present invention to those skilled in the art. Those skilled in the art will appreciate numerous variations from these examples that do not depart from the scope of the invention. Those skilled in the art will also appreciate that various features described below could be combined in various ways to form multiple variations of the invention.

FIG. 5 illustrates a network architecture 500 in an example of the invention. The network architecture 500 comprises a service network 530, a service network 540, and an access network 520. The access network 520 comprises a database system 522, an access server 524, a firewall/router 526, a firewall/router 556, and an access server 554. A user network 510 includes a network device 512. The user network 510 is connected to the access server 524. The access server 524 is connected to the firewall/router 526 and the database system 522. The firewall/router 526 is connected to the service network 530, the service network 540, and the database system 522. The firewall/router 556 is connected to the service network 530, the service network 540, the database system 522, and the access server 554. The access server 554 is connected to the database system 522 and the user network 560. The user network 560 comprises a network device 562 and a network device 564.

The access network 520 provides an interface between the user network 510 and 560 and the service networks 530 and

540. The interface function provides user access profiles, security, switching, and caching. The user network 510 and 560 could be a residential or business communication system that includes network devices. A network device 512, 562, and 564 could be any device configured to exchange data or information with the access network 520. Some examples of network devices are wireless and wireline telephones, computers, modems, servers, and/or data terminals, along with associated interconnections and network interfaces. For illustrative purposes in the examples below, a user interacts with the network device 512 to access services provided by the network architecture 500 and the user network 560. The user network 510 could also be a communication destination for other users. In these cases, the access network 520 provides destination performance monitoring and control. The example of a user interacting with the network device 562 and 564 to access services provided by the network architecture 500 and the user network 510 is not discussed below for the sake of clarity.

The access server 524 and the database system 522 could be adapted from components used in current ISPs, and the access network 520 could be integrated into these ISP components. The database system 522 houses user access profiles for the users in the user networks 510 and 560 with external access rights. The user access profiles provide a data clearinghouse for user-related information for security, service options, current state, and customized macros. The service networks 530 and 540 could be voice or data systems, such as the public telephone network, Internet, public data networks, and private data networks.

When a user requests access to services, the access network 520 processes the user access profile for the user. The access network 520 performs security measures to validate the user. The access network 520 then binds the user to a terminal and to a service. This user-terminal-service binding correlates the user's capabilities in the user access profile with the capabilities of the user's current terminal and the current capabilities of the service provider. The access network 520 activates a network shell for the user and the user's terminal. The network shell is an interface that is customized for the user and the terminal. The user invokes services using the network shell.

The user access profile may contain several macros that can be as simple as address translations or as complex as lengthy computer programs. The user's network shell provides access to the user's macros. The user access profile also stores caching instructions. The caching instructions control the collection and storage of information within the access network 520 for immediate access by the user.

FIG. 6 depicts an access network in an example of the invention. The access network 520 comprises the database system 522, the access server 524, the firewall/router 526, the firewall/router 556, and the access server 554. The database system 522 comprises a local database system 570, a Lightweight Directory Access Protocol (LDAP) interface system 571, a central database system 580, a local database system 590, and an LDAP interface system 591. The local database system comprises a user profile system 572, an audit database system 573, a cache database system 574, a host system 575, a security server 576, a user authorization system 577, an alias translation system 578, and a personal DNS system 579. The central database system 580 comprises a user authorization system 581, a financial interface 582, and a cross connect system 583. The local database system 590 comprises a user profile system 592, a user authorization system 593, and an availability system 594.

The access server 524 and the firewall/router 526 are connected to the local database system 570. The local

database system 570 is connected to the central database system 580 and the LDAP interface system 571. The central database system 580 is connected to the local database system 590. The local database system 590 is connected to the firewall/router 556, the access server 554, and the LDAP interface system 591.

An access provider is an entity that provides access to users who use communications services. A typical access provider comprises an access server, a firewall/router, and a local database system. The systems within the local database system 570 could be included within the local database system 590. Also, the systems within the local database system 590 could be included within the local database system 570. As discussed above, the user is accessing the network architecture through the user network 510. The duplication of systems in the local database system 570 and the local database system 590 is excluded for the purposes of clarity. A service provider is an entity that provides communication services to users who are accessing the service through an access provider.

Network User Access Profile—FIGS. 5, 6, 7, 8, 9, 10, 11, and 12

User Access Profile Inheritance

The user access profile is stored in the access network 520 and controls user access to services. The user access profile is any information or data associated with controlling user access to a service such as role identification, authorization, billing information and access preferences.

FIG. 7 depicts table for a user access profile in an example of the invention. A user access profile includes access information for the user, billing information, and preferences for access. Access information is any information or data related to providing the user access to the network architecture 500. Some examples of access information are user ID, password, name, account number, user alias, current network address, switching allowed flag, and other security information. Access information also include a list of services that the user has subscribed to or is allowed access to. In one embodiment, access information includes a cache of information that the user has accessed previously. Some examples of billing information are address and billing code including bank card numbers or prepaid account codes. Preferences for access allow the user to save choices or preferences to customize their access to the network architecture 400. Some examples of preferences for access are file formats and Quality of Service values. The user access profile may also include usage information such as time of day access, day of week, usages per day, usages per week, and usages per month.

Users typically set up their user access profiles when signing up for the access to the network architecture 500. In one embodiment, users create the user access profile through an inheritance process. Through the inheritance process, the user selects user profile information from other user access profiles in the network architecture 500. The user may inherit user profile information from user profile data structures such as templates, other user access profiles, the user's group or class, or other networks that the user is set up in. A user profile data structure is any user profile information to be retrieved when inheriting a user access profile.

In a prior solution, a command interpreter in a single computer operating system shell environment allows a user to customize the interpretation of command strings. The user may inherit portions of other user's shell environment by adding the other user's shell attributes. User access profiles are typically stored in the access provider's database. FIGS.

5 and FIG. 8 disclose one embodiment for inheriting user access profiles in an example of the invention. The user access profile is created through an inheritance process where the user is able to select capabilities, macros, functions, methods, and data to inherit from other profiles. In this embodiment, users inherit user access profiles from classes, groups, or provider recommendations. Features of user access profiles such as alias translation could then be implemented with new users rapidly. The inheritance of user access profiles also simplifies the configuration of users. In one example, a user initiates the user access profile inheritance by clicking a button on a website. Also, when a user requests a new service, the service provider automatically inherits the user access profile for the user so the user is able to use the requested service.

FIG. 8 depicts a flowchart for the access server 524 for inheriting a user access profile in an example of the invention. FIG. 8 begins in step 800. In step 802, the access server 524 waits for the next packet. The access server 524 then receives and processes the packet from the network device 562 in step 804. The access server 524 then checks if the destination is the user access profile in step 806. If the destination is not the user access profile, the access server 524 transmits the packet on the correct path in step 808 before returning to step 802. If the destination is the user access profile, the access server 524 then checks if the network device 562 is allowed access to the user access profile in step 1110. If the network device 512 is not allowed access to the user access profile, the access server 524 discards the packet and registers a network request security event in step 812 before returning to step 802.

If the network device 512 is allowed access to the user access profile, the access server 524 then checks if the user is allowed to update their user access profile in step 814. If the user is not allowed updates to their user access profile, the access server 524 generates and transmits a profile update not allowed message to the network device 512 in step 816 before returning to step 802. If the user is allowed updates to their user access profile, the access server 524 generates and transmits a user access profile update permission message asking if the user wishes to update the user access profile to the network device 512 in step 818. The access server 524 then checks if the user approved the user access profile update in step 820. If the user does not approve, the access server 524 generates and transmits a user access profile aborted message to the network device 512 in step 822 before returning to step 802.

If the user approves the user access profile update, the access server 524 sets an external request timer in step 824. The access server 524 then generates and transmits a user access profile update request to the database system 522 in step 826. A user access profile update request could be any signaling, message, or indication to update the user access profile. The access server 524 then checks if a reply was received or the external request timer expired in step 828. If the reply was not received and the external request timer did not expire, the access server 524 returns to step 828. If the reply was received or the external request timer did expire, the access server 524 checks if the reply was valid in step 830. If the reply was valid, the access server 524 generates an update complete message in step 832 before returning to step 802. If the reply was not valid, the access server 524 discards the packet and replies to the network device 512 that the user access profile update failed in step 834 before returning to step 802.

FIG. 9 depicts a flow chart for the database system 522 for inheriting a user access profile in an example of the inven-

tion. FIG. 9 begins in step 900. In step 902, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 904. In step 906, the database system 522 then checks if the access server requester is known. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 912. The database system 522 appends the reply to the packet and transmits the reply to the access server 524 in step 910. The database system 522 then checks if the request is a security event in step 916. If the request is not a security event, the database system 522 then registers an unknown action event in step 918 before returning to step 910. If the request is a security event, the database system 522 increments a path/device security record in step 924. The database system 522 then logs the path/device security record in step 926 before returning to step 902.

If the requester is known, the database system 522 then checks whether the request is a user access profile update request in step 914. If the request is not a user access profile update request, the database system 522 checks if the request is a security event in step 916. If the request is not a security event, the database system 522 then registers an unknown action event in step 918 before returning to step 910. If the request is a security event, the database system 522 increments a path/device security record in step 924. The database system 522 then logs the path/device security record in step 926 before returning to step 902.

If the request is a user access profile update request, the database system 522 identifies and authenticates the user and the network device 512 in step 920. The database system 522 then checks if the access is valid in step 922. If the access is invalid, the database system 522 returns to step 924. If the access is valid, the database system 522 retrieves the user access profile information from a user profile data structure in step 928. The retrieval of user access profile information may be based on any information in the user access profile such as group or class or selections of inheritance user access profile presented to the user. The group or classes could be any logical grouping of user based on similar interests or situations such as work, family, and geographic location. The database system 522 then updates the user access profile in step 930 based on the user access profile information retrieved in step 928 and the user's selections for updating. The database system 522 then replies with an approve message to the access server 524 in step 932 before returning to step 902. In one embodiment of the invention, the database system 522 uses the user access profile system 572 to perform the operations disclosed in FIG. 9 and stores the user access profile in the user access profile system 572.

Network Shell

The user access profile includes data to provide a customized network shell to the user. The network shell is a user interface to the access network that is linked to programs, methods, macros, or service. Typically, configuration of the network shell is graphically presented to the user on a display. For example, the network shell appears as a list of alias selections that are associated with actions such as a program or macro for resources or services. An alias selection is any information that is associated with an action to be executed when the alias selection is selected. In another example, the alias selections are graphically presented as icons that relate to actions to be executed. The network shell overlays the standard DNS offered in IP networks, which reduces alias translation delays and required user keystrokes. In prior solutions, a command interpreter in an operating system shell environment allows a user to customize the interpretation of command strings, or a DNS translates "alias" values to addresses. FIGS. 5, 10, and 11 show one embodiment for network shells in an example of the invention.

FIG. 10 depicts a flowchart of an access server for executing a network shell in an example of the invention.

13

FIG. 10 begins in step 1000. In step 1002, the access server 524 waits for the next packet. The user selects an alias selection from the network shell graphically presented. In another embodiment, the user enters an alias value by a non-graphical means. Alternatively, the network shell is not presented to the user. The network device 512 transmits a packet including the alias selection to the access server 524. In step 1004, the access server 524 receives and processes the packet from the network device 512 and processes the packet. The access server 524 then checks if the network device 512 is allowed access to the network architecture 500. If the network device 512 is not allowed access to the network architecture 500, the access server 524 discards the packet and registers a network security event in step 1008 before returning to step 1002.

If the network device 512 is allowed access to the network architecture 500, the access server 524 checks if the user is recognized in step 1010. If the user is not recognized, the access server 524 returns to step 1008. If the user is recognized, the access server 524 retrieves the user's network shell in step 1012. In some embodiments, the user's network shell is retrieved from the user access profile in the access database 522 prior to presenting the network shell to the user. In step 1014, the access server 524 checks if the packet from the network device 512 includes an alias selection from the user's network shell. In one embodiment, specialized hardware is used to scan for aliases just as IP addresses are currently scanned for. If the packet does not include an alias selection from the user's network shell, the access server 524 proceeds to step 1018. If the packet does include the alias selection from the user's network shell, the access server 524 executes the action associated with the alias selection in step 1016 before returning to step 1002. In step 1018, the access server 524 processes the packet with the normal handling before returning to step 1002.

FIG. 11 depicts a flowchart of a database system for updating a network shell in an example of the invention. FIG. 11 begins in step 1120. In step 1122, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 1124. The database system 522 then checks if the access server requester is known in step 1126. If the access server requester is not known, the database system 522 registers an unknown requester event in step 1128. The database system 522 then appends the reply to the packet and transmits the packet to the access server 524 in step 1130. In step 1132, the database system 522 replies with a decline message to the access server 524 before returning to step 1122.

If the access server requester is known, the database system 522 checks if the request is a profile update for the network shell in step 1134. If the request is a profile update for the network shell, the database system 522 identifies and authenticates the user and the network device 512 in step 1136. The database system 522 then checks whether the access is valid in step 1138. If the access is invalid, the database system proceeds to step 1154. If the access is valid, the database system 522 retrieves the user access profile in step 1140. The database system 522 then updates the user access profile with the network shell with the user's alias selections and the associated programs, macros, functions or methods in step 1142. The database system 522 then replies with an approve message in step 1144 before returning to step 1122.

If the request is not a profile update, the database system 522 checks if the request is an action event in step 1146. If the request is an action event, the database system 522

14

performs the action and replies in step 1148 before returning to step 1122. If the request is not an action event, the database system 522 checks if the request is a security event in step 1150. If the request is not a security event, the database system 522 registers an unknown request type event in step 1152 before returning to step 1122. If the request is a security event, the database system 522 proceeds to step 1154. In step 1154, the database system increments a path/device security record. The database system 522 then appends the requester information to the packet and logs the event before returning to step 1122. In one embodiment of the invention, the database system 522 uses the user access profile system 572 to perform the operations disclosed in FIG. 11 and stores the user access profile in the user access profile system 572.

Service Based Directory

In prior systems, the RADIUS server provided the user with selections for transport modes. However, the user was not able to select available network based services. FIGS. 5 and 12 disclose one embodiment for a service based directory in an example of the invention. In this embodiment, access providers provide a list of services that the user can select. With the list of services, the access providers have the ability to advertise specific services to users. The list of services may be generated based on the user access profile to make the list user specific. Once the user makes the selection, the access server 524 connects the user to the service network such as Intranet, Internet, or private dedicated network that provides the selected service.

FIG. 12 depicts a flowchart for the services based directory in an example of the invention. FIG. 12 begins in step 1200. In step 1202, the access server 524 waits for the next connection from a network device in the user network 510. The access server 524 then receives a connection from the network device 512 in step 1204. Once the connection is established, the access server 524 generates and transmits a user ID query to the network device 512 in step 1206. The access server 524 then receives an ID reply and establishes a network device session in step 1208.

The access server 524 then generates an available services reply including a list of services in 1210. In one embodiment, the access server 524 generates the available services reply based upon information in the user access profile. The access server 524 receives a selected service reply from the network device 512 in step 1212. The access server 524 then connects the network device 512 to the selected service provider in step 1214. The access server 524 waits for the next packet in step 1214. The access server 524 then exchanges packets between the network device 524 and the selected service provider in step 1218.

Access Network User Binding—FIGS. 5, 13, 14, 15, and 16

User Access Profile Mobility

Users may access their user access profile from any network device connected to the network architecture 500. In a prior solution, a command interpreter in an operating system shell environment allows a user to customize the interpretation of command strings. In this environment, the access network identifies the user and the terminal device interface, which provides user mobility. The access network then executes customized actions for the user. FIGS. 5 and 13 show one embodiment of the invention for user access profile mobility. This embodiment provides user mobility in a distributed data network environment.

FIG. 13 depicts a flowchart of user access profile mobility in an example of the invention. FIG. 13 begins at step 1300. A user at the network device 512 signs on the access network

520 with their user ID. The network device 512 transmits user information with the user ID to the access server 524. In this embodiment, the user information is in the form of a packet. In step 1302, the access server 524 receives and processes the packet to check if the user access profile is local within the local database system 570. A user access profile is local within the local database system 570 when the user access profile is located in the local database system 570. If the user access profile is local, the access server 524 proceeds to step 1312.

If the user access profile is not local within the local database system 570, the access server 524 checks if the user ID is delimited with a provider ID in step 1304. One example of a user ID delimited with a provider ID includes a user's name and a provider ID separated by a delimiter such as joesmith@access.net. If the user ID is not delimited, the access server 524 retrieves the location of the default user access profile system using a default Lightweight Directory Access Protocol (LDAP) interface system 591 in step 1308 before proceeding to step 1312.

If the user ID is delimited, the access server 524 checks if the provider ID is valid in step 1306. If the provider ID is not valid, the access server 524 registers an invalid profile request security event before terminating at step 1326. If the provider ID is valid, the access server 524 uses the provider ID to retrieve the location of the local database system 590 from a foreign LDAP interface system 591 before proceeding to step 1312.

In step 1312, the access server 524 checks if the packet is a retrieve request for the user access profile. If the packet is a retrieve request, the access server 524 generates and transmits a request to retrieve the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then creates the user session ID and updates the user entries of the user access profile with the user ID and date/time updated in step 1312. The access server 524 then transmits a reply message with the session ID and the location of the LDAP server in step 1314 before terminating at step 1326.

If the packet is not a retrieve request, the access server 524 checks if the packet is a release request in step 1316. If the packet is a release request, the access server 524 retrieves the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then updates the user session ID and updates the user entries of the user access profile with the user ID and date/time updated in step 1318. The access server 524 then transmits a reply message with the session ID in step 1320 before terminating at step 1326.

If the packet is not a release request, the access server 524 checks if the packet is an update request in step 1322. If the packet is not an update request, the access server 524 registers an invalid profile request security event before terminating at step 1326. If the packet is an update request, the access server 524 retrieves the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then updates the user access profile with the information in the packet in step 1324. The access server 524 then transmits a complete message to the user network 510 to signify the profile update is complete before terminating at step 1326. In one embodiment, the local database system 570 uses the user access profile system 572 to retrieve and store the local user access profiles and the local database system 590 uses the user access

profile system 592 to retrieve and store the foreign user access profiles.

User, Device, and Service Sessions

Access network providers provide user and device sessions in addition to service sessions to distinguish users when providing communication services. A user session is the information associated with a user accessing a network. A device session is the information associated with a device being used to access a network. A service session is the information associated with a service being provided over a network. Service providers distinguish users instead of access devices or links. This allows multiple users to share a single access device with each user receiving their own customized or preferred services. Advantageously, service providers establish service access rights and restrictions such as preventing adult content for younger viewers or sharing an access device for business and personal use. Also, user, device, and service sessions allow a service provider to group multiple service providers to provide a composite of services to the user similar to a contractor/sub-contractor relationship.

FIGS. 5, 14, 15, and 16 disclose one embodiment for device, user, and service sessions in an example of the invention. FIG. 14 depicts a logical view of device, user, and service sessions in an example of the invention. Devices 1402, 1404, and 1406 are comprised of session type specific information and session links. Session type specific information include public keys, private keys, and session ID. Session links include user sessions, device sessions, and service sessions. The public keys and private keys are for encryption and decryption of messages. Session ID identifies the session ID for the device. User sessions are user sessions that the device is logically linked to. Service sessions are the service sessions the device is logically linked to. Users 1408, 1410, and 1412 are comprised of public keys, private keys, session ID, device sessions, and service sessions. Session ID identifies the session ID for the user. Device sessions are the device sessions the user is logically linked to. Service sessions are the service sessions the user is logically linked to. Services 1414, 1416, and 1418 are comprised of public keys, private keys, session ID, user sessions, device sessions, and sub service sessions. Session ID identifies the session ID for the service. User sessions are the user sessions the service is logically linked to. Device sessions are the device sessions the service is logically linked to. Sub service sessions are the service sessions the service is logically linked to.

In one example, device 1402 is linked to user B 1410 via a link 1422. User B is also linked to service N 1418 via a link 1424. Device 1402 contains the user information in the user session fields for user B 1410 and the service information in the service session fields for service N. User B 1410 contains the device information in the device session fields for device 1402 and the service information in the service session fields for service N 1418. Service N 1418 contains the device information in the device session fields for device 1402 and the user information in the user session fields for user B 1410. Service N 1418 is also linked to service A 1414 via a link 1426. Service N 1418 contains additional service information in the service session fields for service A 1414. Service A 1414 contains the composite service information in the service session fields for service N 1418. Service A 1414 also includes an owning service which is a reference to the service that owns service A.

FIG. 15 depicts a flowchart for a user based session in an example of the invention. FIG. 15 begins in step 1500. In step 1502, the network device 512 establishes a connection

17

with the access server 524. The access server 524 generates and transmits a user ID query to the user network 510 in step 1504. In step 1506, the network device 512 receives the user ID query and transmits a user ID reply to the access server 524. The access server 524 receives and processes the user ID reply to generate a device session. The network device 512 then transmits a packet to the access server 524. The access server 524 then receives the packet in step 1508. In step 1510, the access server 524 processes the packet to check if the packet includes a user session ID.

If the packet does not include the user session ID, the access server 524 transmits a logon query encrypted by the network device's 512 public encryption key 1402 to the network device 512 in step 1512. The network device 512 decodes the logon query with its public encryption key and transmits a logon reply encrypted with its public encryption key in step 1516. The access server 524 then decodes the logon reply with the private encryption key and checks if the user ID is valid. If the user is valid, the access server 524 generates a user session and a session ID in step 1520. In some embodiments, the user session ID is the original IP address. The access server 524 then encrypts with the public encryption key and transmits a complete reply with the user session ID to the network device 512 in step 1522 before returning to step 1508.

If the packet does include the user session ID, the access server 524 checks if the user and user session ID are valid in step 1514. If the user or user session ID is not valid, the access server 524 discards the packet and registers a security event in step 1515 before returning to step 1508. If the user and user session ID are valid, the access server 524 generates a service request with the user session ID and the service session ID if available. The access server 524 encrypts the service request with the public encryption key where appropriate. The access server 524 transmits the packet including the service request in step 1518 before returning to step 1508.

FIG. 16 depicts a flowchart for a service based session in an example of the invention. FIG. 16 begins in step 1600. The network device 512 transmits a service request to the access server 524. The access server 524 then receives the service request in step 1602. In step 1604, the access server 524 checks if the service request includes a service session ID.

If the service request does not include the service session ID, the access server 524 transmits a service ID query encrypted with the public encryption key to the network device 512 in step 1606. The network device 512 decodes the service ID query with its private key and transmits a service reply encrypted with the service public encryption key to the access server 524 in step 1610. The access server 524 then decodes the service reply with the service private encryption key and checks if the user is valid. If the user is valid, the access server 524 generates a service session and a session ID in step 1614. The access server 524 then transmits a complete reply with the service session ID to the network device 512 in step 1616 before returning to step 1602.

If the packet does include the service session ID, the access server 524 checks if the user and service session ID are valid in step 1608. In one embodiment, the service session ID is the destination IP address. If the user or service session ID is not valid, the access server 524 discards the packet and registers a security event before returning to step 1602. If the user and service session ID is valid, the access server 524 updates the service request with the user session ID and the service session ID. The access server 524

18

encrypts the service request with the service public encryption key where appropriate. The access server 524 transmits the service request with the user session ID and the service session ID in step 1612 before returning to step 1602.

Access Network Security—FIGS. 5, 17, and 18 Service Capability Firewall

A network service provider interfaces with a network access provider at a transport functional level. The interface typically includes Internet protocol firewalls to provide security. Unfortunately, the interface between the network service provider and the network access provider is not able to hide the implementation details such as addressing schemes, transport details, and equipment specifics. The hiding of implementation details is preferred to prevent hackers from manipulating the implementation details. FIGS. 5, 17 and 18 depict one embodiment for a service capability firewall in an example of the invention. In this embodiment, the interface between the network service provider and the network access provider occurs at a named functional level instead of the transport addressing level. A named function request is any request for a capability of service provided by the network service provider. This allows the network service provider to hide the implementation details. The network service provider exposes only functional capabilities to the users depending on their security rights.

FIG. 17 depicts a flowchart for the firewall 556 for service capability firewall in an example of the invention. FIG. 17 begins in step 1700. In step 1702, the firewall 556 waits for information. In this embodiment, the information is in the form of a packet. The firewall 556 receives and processes a packet including a named function request from the network device 512 in step 1704. The firewall 556 then checks whether the firewall 556 is the destination for the named function request in step 1706. If the firewall 556 is not the destination, the firewall 556 registers a network request security event in step 1710. In step 1712, the firewall 556 encapsulates the packet with path information and transmits the packet to the database system 522 before returning to step 1702.

If the firewall 556 is the destination, the firewall 556 checks whether the sending address is consistent with the path in step 1714. If the sending address is not consistent with the path, the firewall 556 returns to step 1710. If the sending address is consistent with the path and does not belong to the accessing network, the firewall 556 checks if the sending address/session ID/named function combination is cached in step 1716. If the sending address/session ID/named function combination is cached, the firewall 556 replaces the packet's named function request with the private cached destination address in step 1718. The firewall 556 then checks if the private destination address is known and allowed to pass in step 1720. If the destination is known and allowed to pass, the firewall 556 transmits the packet on the correct path with standard firewall access in step 1722 before returning to step 3802. If the destination is not known or not allowed to pass, the firewall 556 returns to step 1710.

If the sending address/session ID/named function combination is not cached, the firewall 556 set an external request timer in step 1724. The firewall 556 then generates and transmits a request for the database system 522 in step 1726. The firewall 556 checks whether a reply is received or the timer has expired in step 1728. If the reply is not received and the timer has not expired, the firewall 556 returns to step 1728. In another embodiment, a blocked I/O is used instead of the wait loop in step 1728. If the reply is received or the timer has expired, the firewall 556 checks if there is a valid

reply in step 1730. If the reply is invalid or no reply was received, the firewall 556 discards the packet and registers a translation failure event in step 1734 before returning to step 1702. If the reply is valid, the firewall 556 replaces the packet's named function request based on the reply and caches the address/session functions returned in step 1732 before returning to step 1720.

FIG. 18 depicts a flowchart for the database system 522 for service capability firewall in an example of the invention. FIG. 18 begins in step 1800. In step 1802, the database system 522 waits for the next request. The database system 522 then receives and processes the request in step 1804. In step 1806, the database system 522 then checks if the access server requester is known. If the access server requester is not known, the database system 522 registers an unknown requester event in step 1808. The database system 522 then appends the reply with the packet and replies to the access server 524 in step 1810. The database system 522 then generates and transmits a decline reply to the access server 524 in step 1812 before returning to step 1802.

If the requester is known, the database system 522 then checks if the request is a capability appeal in step 1814. A capability appeal is any message, signaling or instruction requesting a capability with a related network address in the service provider. If the request is not a capability appeal, the database system 522 then checks if the request is a security event in step 1816. If the request is not a security event, the database system 522 register an unknown request event in step 1817 before returning to step 1810. If the request is a security event, the database system 522 increments a path/device security record and appends the requester information to the request packet and logs the event in step 1822 before returning to step 1802. If the request is a capability appeal, the database system 522 then identifies the user and network device 512 in step 1818. The database system 522 then checks if the access is valid in step 1822. If the access is invalid, the database system 522 returns to step 1822. If the access is valid, the database system 522 retrieves the user and network device profiles in step 1824. The database system 522 then checks whether the capability is valid for the user and session state in step 1826. If the capability is invalid, the database system 522 returns to step 1822. If the capability is valid, the database system 522 generates a session ID, the network address of the capability requested, and functions available and appends the network address, which is only sent to the firewall/router 526 and not to the user, to the reply in step 1830. The database system 522 then transmits the approve reply to the access server 524 in step 1832 before returning to step 1802.

Access Network Service Authorization—

FIGS. 5, 19, 20, 21, 22, 23, and 24

Prepaid Access

Prepaid phone cards are commonly used in PSTN, where the customer pays a prepaid amount that is debited against when the customer makes a call. In data networks, prepaid cards are not currently being used. FIGS. 5 and 19 disclose one embodiment for prepaid access in an example of the invention. In this embodiment, users buy prepaid cards from network providers. When the user requests access to one of many access providers throughout the country, the access provider verifies the prepaid account code before providing the access. A prepaid account code is any number that relates to a user's prepaid account. The prepaid account is debited against for the charges related to the access. Other charges related to the service provided may also be debited against the prepaid account. For example, a user may purchase an item from a website and have the charges debited against the

prepaid account. Once the prepaid amount is reached, the access provider terminates the access to the user. In one embodiment, the network provider provides different levels of service such as gold, silver, and bronze. The gold service has guaranteed throughput but higher rates for access, while the bronze service has lower throughput and rates.

FIG. 19 depicts a flowchart for prepaid access in an example of the invention. FIG. 19 begins in step 1900. In step 1902, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 1904. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits information identifying a billing code. A billing code is any number that identifies a user for billing. Two examples of billing codes are prepaid account codes and credit card numbers. In this embodiment, the information identifying a billing code is a response including a prepaid account code to the login query. In other embodiments, the billing code is retrieved from the user access profile. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response including the prepaid account code to determine if the user is known in the local database system 520 in step 1906. A user is known when the user is allowed to use the access network 520. In one embodiment, the user is known in the local database system 520 if there is time/amounts left in the user's prepaid account. In another embodiment, the database system 522 evaluates a positive balance file to determine the remaining time/amount in the user's prepaid account. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 1908 before returning to step 1902.

If the user is not known, the database system 522 checks if there is an appeal server for user authentication in step 1910. In one embodiment, the access server 524 performs the appeal on a decline. If there is no appeal server, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2412 before returning to step 2402. If there is an appeal server, the database system 522 generates an authorization query including the prepaid account code for the central database system 580 in step 1914. The database system 522 checks if the user is known in the central database system 580 in step 1914. In one embodiment, the user is known in the central database system 580 if there is time/amounts left in the user's prepaid account. If the user is known, the database system 522 proceeds to step 1908. If the user is not known, the database system 522 proceeds to step 1910.

In one embodiment, the database system 522 uses a user authorization system 575 for checking if the user is known in the local database system 520. The user authorization system 575 contains all the prepaid customers, prepaid customer information, prepaid account codes, and the amount/quantity remaining in the prepaid account to verify if access is allowed. In one embodiment, the database system 522 uses a user authorization system 581 as the appeal server for checking if the user is known in the local database system 580. The user authorization system 581 contains all the prepaid customers, prepaid customer information, and the amount/quantity remaining in the prepaid account.

Bank Card Access

In prior systems, access providers authenticated users using their own databases. FIGS. 5, 20 and 21 disclose one

embodiment for bank card access in an example of the invention. In this embodiment, access providers authenticate users through bank card financial networks using the users' credit card numbers. Network providers use credit or debit card numbers as user ID and passwords for authentication and authorization purposes. Users use prepaid cards, phone cards, and credit cards in PSTN. However, no bank cards have been used for access to data networks other than for batch bill payment.

FIG. 20 depicts a flowchart for bank card access for a connection in an example of the invention. FIG. 20 begins in step 2000. In step 2002, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2004. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits information identifying a billing code. In this embodiment, the information identifying a billing code is a response including credit card numbers to the login query. In other embodiments, the billing code is retrieved from the user access profile. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response to see if the user is known in the local database system 570 in step 2006. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 2008 before returning to step 2002.

If the user is not known, the database system 522 processes the response to check if the user is identified by credit card numbers in step 2010. If the user is not identified by the credit card numbers, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2012 before returning to step 2002. If the user is identified by the credit card numbers, the database system 522 generates an authorization query as a pre-authorization hold or authorization/capture transaction for the central database system 580 in step 2014. In one embodiment, the central database system 580 uses a financial interface 582 to interface with a financial switch to banks for authentication and authorization. The database system 522 then checks if the user is authenticated and authorized by the central database system 580 in step 2016. If the user is authenticated and authorized, the database system 522 logs access information and an authorizing financial entity or institution in step 2018 before proceeding to step 2008. If the user is not known, the database system 522 checks if there is another database system for authorization such as for foreign user access in step 2018. If there is another database system, the database system 522 returns to step 2014. If there is not another database system, the database system 522 returns to step 2012.

FIG. 21 depicts a flowchart for bank card access for a disconnection in an example of the invention. FIG. 21 begins in step 2100. In step 2102, the access server 524 waits for a disconnection. The network device 512 then disconnects from the access server 524 in step 2104. The access server 524 then checks whether the user was on a pre-authorization hold in step 2106. If the user is not on a pre-authorization hold, the 522 logs access information and an authorizing database system 522 in step 2108 before proceeding to step 2102. If the user is on a pre-authorization hold, the database system 522 generates a pre-authorization complete transaction for the central database system 580 in step 2110 before returning to step 2108.

Access Cards

In prior systems, access providers authenticated users using their own databases. FIG. 5, 22 and 23 disclose one embodiment for network access cards in an example of the invention. An access card is a card that a user uses to access a network. The access card includes an access card account code. The access card account code is any number that relates to the user's access account. In this embodiment, access providers authenticate users who have access cards using other access providers' databases. Access providers then may enter into sharing agreements with other access providers. These sharing agreements allow users to use multiple access providers, which provide the users greater mobility and availability. Users use phone cards in PSTN for phone card access. However, no access cards for data networks have been used.

FIG. 22 depicts a flowchart for network access cards for a connection in an example of the invention. FIG. 22 begins in step 2200. In step 2202, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2204. In another embodiment of the invention, the user calls a toll free number to request access. A service control point is queried to determine where to route the request for access similar to an automatic call distribution (ACD). The request for access is then routed to the access server 524 to establish a connection between the network device 512 and the access server 524.

Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits a response including the access card account code to the login query. The access server 524 then transfers the response from the network device 512 to the database system 522. In this embodiment, the database system 522 checks if the user is allowed to use the access network 520 by checking whether the user is known. A user is known when the user is allowed to use the access network 520. For example, a user is known when their user access profile is located within the local database system 570 and their user access profile allows the user to use the access network 520. The database system 522 receives and processes the response including the access card account code to check if the user is known in the local database system 570 in step 2206. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 2208 before returning to step 2202.

If the user is not known, the database system 522 checks if the response included foreign network account information in step 2210. Foreign network account information is any information that is indicative of an account that is external to local database system 570 that the user is attempting to gain access. If there is no foreign network account information, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2212 before returning to step 2202.

If there is foreign network account information, the database system 522 identifies the local database system 590 based on the foreign network account information and generates an authorization query for the local database system 590 in step 2214. The database system 522 then checks if the user is authenticated and authorized by the local database system 590 in step 2216. If the user is authenticated and authorized, the database system 522 logs contract and settlements information returned by the local database system 590 or indicated by the database system 522 in relation to local database system 590 in step 2218 before proceeding to step 2208. If the user is not known, the

23

database system 522 proceeds to step 2212. In one embodiment, the local database system 570 uses the user authorization system 575 to check if the user is known in the local database system 570. In one embodiment, the local database system 590 uses the user authentication system 593 for authentication and authorization.

FIG. 23 depicts a flowchart for network access cards for a disconnection in an example of the invention. FIG. 23 begins in step 2300. In step 2302, the access server 524 waits for a disconnection. The network device 512 then disconnects from the access server 524 in step 2304. The access server 524 then generates and transmits a logoff query for the database system 522. The database system 522 then transfers the logoff query to the local database system 570 and the local database system 590. The database system 522 logs the access information and the authorizing database system in step 2308 before returning to step 2302.

Global Authentication

In prior systems, access providers authenticated users using their own databases. FIGS. 5 and 23 disclose one embodiment for global authentication in an example of the invention. In this embodiment, access providers authenticate users using a centralized database. Access providers then may enter into sharing agreements with other access providers. These sharing agreements allow users to use multiple access providers, which provide the users greater mobility. Currently, cellular telephone companies enter into these sharing arrangements for greater coverage. However, this sharing of databases for authentication and authorization has not occurred for data networks.

FIG. 24 depicts a flowchart for global access in an example of the invention. FIG. 24 begins in step 2400. In step 2402, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2404. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits a response to the login query. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response to see if the user is native in the local database system 570 in step 2406. In this embodiment, the database system 522 checks if the user is allowed to use the access network 520 by checking whether the user is native. A user is native when the user is allowed to use the access network 520. For example, a user is native when their user access profile is located within the local database system 570 and their user access profile allows the user to use the access network 520. If the user is native, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 4308 before returning to step 2402.

If the user is not native, the database system 522 checks if there is an authentication/authorization server in the database system 522 for a foreign network for user authentication in step 1910. If there is no authentication/authorization server in the database system 522 for the foreign network, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2412 before returning to step 2402. If there is an authentication/authorization server in the database system 522 for the foreign network, the database system 522 generates an authorization query for the central database system 580 in step 2414. The database system 522 then checks if the user is known in the central database system 580 in step 2414. If the user is known, the database

24

system 522 proceeds to step 2408. If the user is not known, the database system 522 proceeds to step 2410. In one embodiment, the central database system 580 uses a user authorization system 581 to check if the user is known.

Access Network Proxy/Environment—
FIGS. 5, 25, 26, 27, 28, 29, 30, and 31
User Based Proxy

A proxy is an application that represents itself as one or more network endpoints. The proxy receives a request for services from a user at a network endpoint and acts on behalf of the user in transmitting and receiving user requests and replies. There are Internet proxy agents that are user network access bind points. However, the Internet proxy agents are not user specific, and they act to protect user interests, not network. These proxy agents provide address translation and basic firewall functionality. Client focused proxies have extended to cookie collection and password handling.

FIGS. 5, 25 and 26 disclose one embodiment for user based proxies in an example of the invention. The user based proxies obtain information for the user and establishes a user specific network presence. A benefit of having a user specific network presence is that user access is handled by a process owned by a network security certificate authority that prevents Trojan horse network attacks. User based proxy provides a single control and monitor point for a user. The proxy agents provides a bind point for all user specific access such as user profile functionality, translations, security, and caching.

FIG. 25 depicts a flowchart for the access server 524 for user based proxies in an example of the invention. FIG. 25 begins in step 2500. In step 2502, the access server 524 waits for the next event. If the next event is a connection, the network device 512 connects with the access server 524 in step 2504. In step 2506, the access server 524 then collects user information and transfers the user information to the database system 522 to perform the security. In one embodiment, the database system 522 uses a security server 576 to perform the security. The access server 524 then checks if the user is allowed access in step 2508. If the user is not allowed access, the access server 524 disconnects the user in step 2510 before returning to step 2502.

If the user is allowed access, the access server 524 checks if a proxy is available in the database system 522 in step 2512. In one embodiment of the invention, the user proxies are in the host system 575. If the proxy is available, the access server 524 proceeds to step 2516. If the proxy is not available, the access server 524 starts the proxy in the database system 522 in step 2514 before proceeding to step 2516.

The access server 524 checks if the user access profile information is available in step 2516. If the user access profile information is not available, the access server 524 generates and transmits a user profile error to the network device 512 in step 2518 before returning to step 2502. If the user profile information is available, the access server 524 configures the proxy for the user in step 2520. The access server 524 then generates and transmits a message with the address of the user proxy and the public encryption key to the network device 512 in step 2522. The access server 524 also collects and transmits the access address of the network device 512 and the public encryption key to the user proxy. The access server 524 then returns to step 2502.

If the next event is a disconnection, the network device 512 disconnects with the access server in step 2524. The access server 524 then generates and transmits a reset command to the user proxy to clear the proxy state information and configuration in step 2526 before returning to

step 2502. If the next event is a status/request event from the user proxy, the access server 524 sets a user proxy reply timer in step 2528. The user proxy has written a status to the access server 524, and the user proxy receives a reply in step 2530 before returning to step 2502. If the next event is a timer expiration, the access server 524 receives the user proxy reply timer expiration in step 2532. The access server 524 then generates and transmits a continue wait message and status to the user proxy in step 2534 before returning to step 2528.

FIG. 26 depicts a flowchart for a user proxy for user based proxies in an example of the invention. FIG. 26 begins in step 2600. In step 2602, the user proxy waits for the next event. If the next event is the completion of the initialization, the user proxy checks if the initialization is complete in step 2604. The user proxy then sets a request timer in step 2606. The user proxy then request work from the access server 524 and transmits a status to the access server 524 in step 2608 before returning to step 2602. If the next event is the expiration of the request timer, the user proxy checks if the request timer expired in step 2610. The user proxy then registers a timeout in step 2612 before returning to step 2606.

If the next event is an access server 524 reply, the user proxy receives the access server 524 reply in step 2614. The user proxy then checks if the reply is a continue in step 2616. If the reply is a continue, the user proxy returns to step 2606. If the reply is not a continue, the user proxy processes the configuration information or action in step 2618 before returning to step 2606. If the next event is user request, the user proxy receives the user request for service in step 2620. The user proxy then checks if the user is valid in step 2622. If the user is valid, the user proxy then checks if the request is valid in step 2624. If the request is valid, the user proxy packages the request with the security certification and encryption in step 2626. The user proxy then transmits the request to the appropriate network destination in step 2628 before returning to step 2602. If the user or request is not valid, the user proxy registers a security event in step 2630 before returning to step 2602. The user proxy exchanges user information between the user network 510 and the appropriate network destination.

Subscriber Based Proxy

Another type of proxy is a subscriber based proxy. A subscriber is a logical entity such as an organization, corporation, or other grouping of users that has subscribed to services with a service provider. FIGS. 5, 27 and 28 disclose one embodiment for subscriber based proxies in an example of the invention. The subscriber based proxies obtain information for a user of a subscriber group and establishes a subscriber specific network presence. The benefit of having a subscriber specific network presence is that user access rights of the subscriber can be handled as a group, and group rights are owned by a network security certificate authority. The subscriber proxy agents provide a bind point for all subscriber specific access such as user profile functionality, translations, security, and caching.

FIG. 27 depicts a flowchart for the access server 524 for subscriber based proxies in an example of the invention. FIG. 27 begins in step 2700. In step 2702, the access server 524 waits for the next event. If the next event is a connection, the network device 512 connects with the access server 524 in step 2704. In step 2706, the access server 524 then collects user information and transfers the user information to the database system 522 to perform the security. In one embodiment, the database system 522 uses a security server 574 to perform the security. The access server 524

then checks if the user is allowed access in step 2708. If the user is not allowed access, the access server 524 disconnects the user in step 2710 before returning to step 2702. If the user is allowed access, the access server 524 checks if all required subscriber proxies are available in the database system 522 in step 2712. In one embodiment of the invention, the subscriber proxies are in the host system 575. If the proxies are available, the access server 524 proceeds to step 2716. If the proxies are not available, the access server 524 starts the required proxies in the database system 522 in step 2714 before proceeding to step 2716.

In step 2716, the access server 524 checks if the subscriber access profile information is available. If the subscriber access profile information is not available, the access server 524 generates and transmits a subscriber profile error to the network device 512 in step 2718 before returning to step 2702. If the subscriber profile information is available, the access server 524 configures the subscriber proxies for the subscriber in step 2720. The access server 524 then generates and transmits a message with the address of the subscriber proxy and the public encryption key to the network device 512 in step 2722. The access server 524 also collects and transmits the access address of the network device 512 and the public encryption key to the subscriber proxies. The access server 524 then returns to step 2702.

If the next event is a disconnection, the network device 512 disconnects with the access server in step 2724. The access server 524 then generates and transmits a reset command to the subscriber proxy in step 2726 before returning to step 2702. If the next event is a status/request event from the subscriber proxy, the access server 524 sets a subscriber proxy reply timer in step 2728. In step 2730, the subscriber proxy has written a status to the access server 524, and the subscriber proxy receives a reply before returning to step 2702. If the next event is a subscriber proxy timer expiration, the access server 524 receives the subscriber proxy reply timer expiration in step 2732. The access server 524 then generates and transmits a continue wait message and a status to the subscriber proxy in step 2734 before returning to step 2730.

FIG. 28 depicts a flowchart for a subscriber proxy for subscriber based proxies in an example of the invention. FIG. 28 begins in step 2800. In step 2802, the subscriber proxy waits for the next event. If the next event is the completion of the initialization, the subscriber proxy checks if the initialization is complete in step 2804. The subscriber proxy then sets a request timer in step 2806. The subscriber proxy then request work from the access server 524 and transmits a status to the access server 524 in step 2808 before returning to step 2802. If the next event is the expiration of the request timer, the subscriber proxy checks if the request timer expired in step 2810. The subscriber proxy then registers a timeout in step 2812 before returning to step 2806.

If the next event is an access server 524 reply, the subscriber proxy receives the access server 524 reply in step 2814. The subscriber proxy then checks if the reply is a continue in step 2816. If the reply is a continue, the subscriber proxy returns to step 2806. If the reply is not a continue, the subscriber proxy processes the configuration information or action in step 2818 before returning to step 2806. If the next event is a user request for service, the subscriber proxy receives the user request in step 2820. The subscriber proxy then checks if the user is valid in step 2822. If the user is valid, the subscriber proxy then checks if the request is valid in step 2824. If the request is valid, the subscriber proxy packages the request with the security

certification and encryption in step 2826. The subscriber proxy then transmits the request to the appropriate network destination in step 2828 before returning to step 2802. If the user or request is not valid, the subscriber proxy registers a security event in step 2830 before returning to step 2802. The subscriber proxy exchanges user information between the user network 510 and the appropriate network destination.

Dynamic Proxies

One prior system named "pluxy" allows a manual extension of a proxy. Pluxy requires manual determination and loading of a dynamic set of services, which is done on an operator basis. FIGS. 5 and 29 disclose one embodiment for dynamic proxies in an example of the invention. In this embodiment, both the user based proxies and the subscriber based proxies are extended in response to user actions. The dynamic proxy can be modified and enhanced to evolve with new services and/or protocols. The dynamic proxy size is smaller and more efficient by supporting only the logic that is being used. Implementation and development times of new service and protocol are also reduced because new proxies are not required with the new service and/or protocol.

FIG. 29 depicts a flowchart for a dynamic proxy for dynamic proxies in an example of the invention. FIG. 29 begins in step 2900. In step 2902, the dynamic proxy waits for the next event. If the next event is the completion of the initialization, the dynamic proxy checks if the initialization is complete in step 2904. The dynamic proxy then sets a request timer in step 2906. The dynamic proxy then request work from the access server 524 in step 2908 before returning to step 2902. If the next event is the expiration of the request timer, the dynamic proxy checks if the request timer expired in step 2910. The dynamic proxy then registers a timeout in step 2912 before returning to step 2906.

If the next event is an access server 524 reply, the dynamic proxy receives the access server 524 reply in step 2914. The dynamic proxy then checks if the reply is a continue in step 2916. If the reply is a continue, the dynamic proxy returns to step 2906. If the reply is not a continue, the dynamic proxy processes the configuration information in step 2918 before returning to step 2902. If the next event is a user request, the dynamic proxy receives the user request in step 2920. The dynamic proxy then checks if the user and the request are valid in step 2922. If the user and the request are valid, the dynamic proxy checks if there is a new service or protocol requested in step 2924. If there is a new service or protocol, the dynamic proxy proceeds to step 2934. If there is not a new service or protocol, the dynamic proxy packages the request with the security certification and encryption in step 2926. The dynamic proxy then transmits the request to the appropriate network destination in step 2928 before returning to step 2902. If the user or request is not valid, the subscriber proxy registers a security event in step 2930 before returning to step 2902.

If the next event is a new service/protocol request, the dynamic proxy receives the new service/protocol request in step 2932. In step 2934, the dynamic proxy processes the new service/protocol request to generate a handler request for a new service/protocol handler in step 2934 before returning to step 2926. The dynamic proxy receives the handler and executes the handler for the new service or proxy. In one embodiment, the dynamic proxy is enhanced by loading apple-like extensions from a handler system similar to a web browser.

Access Execution Environment

Proxy servers in the Internet represents a user in one network as existing in another network to obtain services for

the user. One problem with Internet proxy servers is the absence of the management of network resource utilization. In TINA-C, the User Agent and the User Session Manager are restricted to known service binding because any new service requires knowledge of or logic for a new CORBA interface. One problem with TINA-C is the absence of any provider resource management capabilities. Unfortunately, both Internet proxy servers and TINA-C do not provide any means of associating or viewing all network resources used by a user access session. Another problem is the lack of securing provider network resource against tampering by accessing users. Also, Internet proxy servers and TINA-C do not provide a platform-supported means of enforcing security levels on network access users. Both system also lack a validation of potential execution capability extensions.

FIGS. 5, 30, and 31 disclose one embodiment for an access execution environment in an example of the invention. The access execution environment easily and efficiently manages and secures network access by providing an execution environment in the access provider environment. Users access shared services and resources through their specific access execution environment. Because the access execution environment is in the access provider environment, the access provider can view the all of the user's activity by observing the execution environment. In this embodiment, the execution environment is setup as a standard system user on the network access platform, which allows management of the system user by operating system level user resource controls, quotas, and management mechanisms.

The access execution environment includes a multi-tasking virtual machine, a script interpreter, alias resolution capabilities, security certificate authentication, configuration handler, session caching, and protocol handling components. FIG. 30 depicts a block diagram of the access execution environment in an example of the invention. In FIG. 30, a network access platform 3000 comprises an execution environment manager 3010, a program and attribute database 3020, an execution environment A 3030, a transport A 3032, an execution environment B 3040, a transport B 3042, an execution environment C 3050, a transport C 3052, an execution environment D 3060, a transport D 3062, an execution environment E 3070, and a transport E 3072. In one embodiment, the network access platform 3000 is included within the database system 522. In another embodiment, the network access platform is included within the host system 575. The transport A 3032 is connected to the execution environment A 3030. The transport B 3042 is connected to the execution environment A 3040 and a workstation 3080. The transport C 3052 is connected to the execution environment C 3050. The transport D 3062 is connected to the execution environment D 3060. The transport E 3072 is connected to the execution environment E 3070. The execution environment A 3030 includes a transport handler 3034, a configuration handler 3036, and a security handler 3038. The execution environment B 3040 includes a transport handler 3044, a configuration handler 3046, and a security handler 3048. The execution environment C 3050 includes a transport handler 3054, a configuration handler 3056, and a security handler 3058. The execution environment D 3060 includes a transport handler 3064, a configuration handler 3066, and a security handler 3068. The execution environment E 3070 includes a transport handler 3074, a configuration handler 3076, and a security handler 3078.

In operation, a system administrator sets up a number of network access user accounts on the hardware platform. The administrator then sets up resource limits for each account/

platform resource. The administrator then starts an execution environment **3030**, **3040**, **3050**, **3060**, and **3070** for each access user account ID. The execution environment **A 3030** initializes and loads a configuration handler **3036**. The execution environment **A 3030** then loads the security handler **3038** and the transport handler **3034**. The transport handler **3034** opens a logical or physical transport port **A 3032** on the hardware platform using port information from the configuration handler.

FIG. 31 depicts a flow chart for the access execution environment in an example of the invention. FIG. 31 begins in step **3100**. In step **3102**, the security handler **3038** waits for a connect message from the transport handler **3034**. The security handler **3038** then receives the connect message from the transport handler **3034** in step **3104**. The security handler **3038** then generates and transmits a logon request via the transport handler **3034** in step **3106**. The security handler **3038** then checks if a reply for the logon request from the transport handler **3034** has been received in step **3108**. If the reply for the logon request has not been received, the security handler **3038** checks if a reply timeout has occurred in step **3110**. If a reply timeout has not occurred, the security handler **3038** returns to step **3108**. If a reply timeout has occurred, the security handler **3038** generates and transmits a disconnect message to the transport handler **3034** in step **3112** and returns to step **3102** to wait for a connect message.

In step **3114**, the security handler **3038** receives the reply for the logon request. Then the security handler **3038** retrieves the location of a security server from the configuration handler **3036** and transmits the logon information to the security server in step **3116**. The security handler **3038** then checks if a reply to the logon information from the security server has been received in step **3118**. If no reply has been received, the security handler **3038** checks if a reply timeout has occurred in step **3120**. If a reply timeout has not occurred, the security handler **3038** returns to step **3118**. If a reply timeout has occurred, the security handler **3038** checks if the logon information has been sent three times in step **3122**. The number of tries could be configurable. If the logon information has not been sent three times, the security handler **3038** returns to step **3116** to transmit the logon information. If the logon information has been sent three times, the security handler **3038** returns to step **3112** to send a disconnect message.

If the security server replied before the reply timeout, the security handler **3038** checks if the reply is an accept message in step **3124**. If the reply is a decline message, the security handler **3038** returns to step **3112**. If the reply is an accept message, the security handler **3038** transmits the accept message configuration parameters to the configuration handler **3036** in step **3126**. The configuration handler **3036** then loads attributes and programs and executes programs specified by the security server reply in step **3128**. The execution environment **A 3030** performs the execution of programs for the user in step **3130**. In another embodiment, the programs request attributes and/or programs to be loaded and/or executed. The programs include the ability to interface with users via the transport handler and load/execute other programs required by request types.

The configuration handler **3036** then checks if the transport handler **3034** receives a disconnect message in step **3132**. If the transport handler **3034** has not received a disconnect message, the configuration handler **3036** returns to step **3130**. If the transport handler **3034** has received a disconnect message, the transport handler **3034** transmits the disconnect message to the configuration handler **3036** in step

3134. In step **3136**, the configuration handler **6536** based on port configuration attributes closes the transport port **3032**, resets the transport handler **3034**, and notifies the execution environment manager **3010** to create a new execution environment for that port. The configuration handler **3036** gracefully shuts down all handlers, programs, and eventually the execution environment **A 3030**. The configuration handler **3036** transmits a shutdown message to the execution environment manager **3010** in step **3138** before terminating in step **3140**. The execution environment manager **3010** restarts the execution environment **A 3030** upon notification of the shutdown. The operations of the other execution environments **3040**, **3050**, **3060**, and **3070** perform in the same manner as the execution environment **A 3030** and are not discussed for the sake of clarity.

Access Network Translations—FIGS. 5, 32, 33, 34, 35, 36, 37, 38, and 39

Domain Name Scoping

The typing in of domain names such as www.nypostonline.com is quite cumbersome for the user to access specific services. The Domain Name Server (DNS) translates the domain name to a network address for the service. Clicking on bookmarks or favorites in browsers, which reference the domain names, does eliminate the need to type in the domain names. However, when users change network devices, these bookmarks or favorites do not follow the user. In an operating system environment, the command interpreter shell allows the user to customize interpretation of command strings. If no customization is loaded, the command interpreters interprets in a default fashion.

FIGS. 5, 32, and 33 disclose one embodiment for domain name scoping in an example of the invention. This embodiment provides a customizable network shell to overlay the standard DNS service offered in Internet Protocol based networks. Users then are able to define user specific acronyms or aliases for specific or logical services. An acronym or alias indicates domain names, macros, programs, actions, or network addresses. If the translation for the alias does not exist in the list of aliases for the user, the access server **524** checks if the alias exists in the list for a group in which the user belongs. The access server **524** then checks if the alias exist in the DNS for the general network. In another embodiment, the database system **522** checks the existence of the alias in the list for the group and in the DNS for the general network. There are numerous variations in the hierarchy for searching for an alias but are not discussed for the sake of simplicity.

FIG. 32 depicts a flowchart for the access server **524** for domain name scoping in an example of the invention. FIG. 32 begins in step **3200**. In step **3202**, the access server **524** waits for a packet. In this embodiment, the access server **524** receives information in the form of packets from the network device **512**. The access server **524** then receives and processes the packet from the network device **512** in step **3204**. The access server **524** then checks if the packet is an alias translation request in step **3206**. If the packet is not an alias translation request, the access server **524** transmits the packet with standard access logic in step **3214** before returning to step **3202**.

If the packet is an alias translation request, the access server **524** then checks if the alias translation is cached for the user in step **3208**. If the alias translation is cached, the access server **524** reformats the packet using the cached alias translation in step **3216** before returning to step **3214**. If the alias translation is not cached, the access server **524** sets an external request timer in step **3210**. The access server **524** then generates and transmits an alias translation request to

the database system 522 in step 3212. The alias translation request is any message, instruction, or signaling indicative of requesting an alias translation. The access server 524 checks whether a reply has been received or the external request timer has expired in step 3218. If a reply has not been received and the external request timer has not expired, the access server 524 returns to step 3218.

If a reply has been received or the external request timer has expired, the access server 524 checks if the reply was valid in step 3220. If the reply is invalid, the access server 524 discards the packet and replies with an invalid alias message or alias error message to the network device 512 before returning to step 3202. If the reply is valid, the access server 524 caches the alias translation for the user in step 3222 before returning to step 3216.

FIG. 33 depicts a flowchart for the database system 522 for domain name scoping in an example of the invention. FIG. 33 begins in step 3300. In step 3302, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 3304. In step 3306, the database system 522 then checks if the access server requester is known in step 3306. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 3308. The database system 522 appends the reply information to the packet in step 3310. The database system 522 replies with a decline message to the access server 524 in step 3312 before returning to step 3302.

If the access server requester is known, the database system 522 then checks whether the request is an alias translation request in step 3314. If the request is not an alias translation request, the database system 522 registers an unknown request event in step 3316 before returning to step 3310. If the request is an alias translation request, the database system 522 identifies and authenticates the user and the network device 512 in step 3318. If the access is invalid, the database system 522 increments a path/device security record in step 3322. The database system 522 then appends the requester denial information to the request packet in step 3324. The database system 522 then replies with a decline message to the access server 524 in step 3312 before returning to step 3302.

If the access is valid, the database system 522 retrieves the user's alias translation list in step 3326. The database system 522 then translates the alias for the user and appends the translation to a reply to the access server 524. The database system 522 then replies with an approve message to the access server 524 in step 3330. In one embodiment of the invention, the database system 522 uses the alias translation system 578 to perform the operations disclosed in FIG. 33 and stores the aliases and alias translations in the alias translation system 578.

Inband Domain Name Service Lookup

The Domain Name Server (DNS) translations of a domain name to a network address delays user interaction with the service provider. In prior solutions in voice telephone Signaling System #7 networks, a signal transfer point performs an inband local number portability (LNP) lookup on a ISDN part call setup request such as an Initial Address Message (IAM). This inband lookup eliminates the network switch from launching a Transaction Capabilities Application Part (TCAP) LNP query to translate the telephone number in the IAM.

FIGS. 5 and 34 disclose one embodiment for inband domain name service in an example of the invention. This embodiment provides a cache for the access server 524 for alias translations on a per user basis. Thus, the external

queries for translations of aliases to domain names, macros, programs, or network addresses are eliminated. If the translated value of a request is in the alias translation cache, no translation request will be required to a DNS server. Therefore, users experience reduced delays when requesting a service.

FIG. 34 depicts a flowchart for the access server 524 for an inband domain name service lookup in an example of the invention. FIG. 34 begins in step 3400. In step 3402, the access server 524 waits for a packet. The access server 524 then receives and processes the packet from the network device 512 in step 3404. The access server 524 then checks if the destination network address and user is valid in step 3406. If the destination network address and user is valid, the access server 524 transmits the packet on the correct path to reach the destination in step 3408 before returning to step 3402.

If the destination network address or the user is invalid, the access server 524 then checks if the alias translation is cached for the user in step 3410. If the alias translation is cached, the access server 524 reformats the packet using the cached alias translation in step 3412 before returning to step 3402. If the alias translation is not cached, the access server 524 sets an external request timer in step 3414. The access server 524 then generates and transmits an alias translation request to the database system 522 in step 3416. The access server 524 checks whether a reply has been received or the external request timer has expired in step 3418. If a reply has been received and the external request timer has not expired, the access server 524 returns to step 3418.

If a reply has been received or the external request timer has expired, the access server 524 checks if the reply was valid in step 3420. If the reply is invalid, the access server 524 discards the packet and replies with an invalid alias message to the network device 512 in step 3424 before returning to step 3402. If the reply is valid, the access server 524 caches the alias translation for the user in step 3422 before returning to step 3412.

Inline Access Service Triggering

Typical firewalls filter packets out on a request by request basis on "what is not allowed". FIGS. 5 and 35 disclose one embodiment for inline access service triggering. In this embodiment, the access server 554 filters packets out on a "what is allowed" basis. Also, the access server 554 performs packet filtering on a service access basis.

FIG. 35 depicts a flowchart for inline access service triggering in an example of the invention. FIG. 35 begins in step 3500. In step 3502, the access server 554 waits for the next packet. In this embodiment, the access server 554 receives information in the form of packets from the network device 512. The access server 554 receives and processes a packet from the network device 512 in step 3504. In step 3506, the access server 554 checks if the protocol, sending address, and the destination address are known and allowed to pass. In one embodiment, the access server 554 checks the database system 522 for the determination made in step 3506. If the protocol, sending address, and the destination address are known and allowed to pass, the access server 554 checks if the sending address is consistent with the path in step 3508. If the sending address is not consistent with the path, the access server 554 registers a path security event in step 3510. The access server 554 then formats the packet with path information in step 3512. The access server 554 also increments the path/device security record and logs the event before returning to step 3502. If the sending address is consistent with the path, the access server 554 transmits the packet on the correct path with standard firewall access in step 3514 before returning to step 3502.

If the protocol, sending address, or the destination address are not known or not allowed to pass, the access server 554 checks if the request is a filter appeal in step 3515. If the request is not a filter appeal, the access server 554 discards the packet in step 3526 before returning to step 3502. If the request is a filter appeal, the access server 554 identifies and authenticates the user and the network device 512 in step 3516. The access server 554 then checks if the access is valid in step 3518. If the access is invalid, the access server 554 returns to step 3510. If the access is valid, the access server 554 retrieves the user access profile and network device's 512 profile in step 3520. The access server 554 then modifies the access logic for filter modification in step 3524. The access server 554 then modifies the protocol and address filters based on the request in step 4824 before returning to step 3514. In some embodiments, the access server 554 modifies the filters in conformance with the user access profile.

Access Service Triggering

Access providers sometimes need to extend their authentication logic beyond their primary access devices. No prior system in data networks extends the authentication logic beyond what is performed by the access provider's access devices. FIGS. 5, 36 and 37 disclose one embodiment for access service triggering. In this embodiment, the access server 524 triggers a request to external access control logic. Access providers extend the access and authentication logic beyond their access devices while maintaining centralized control of the authentication logic and user access profiles. Also, the access server 554 performs packet filtering on a service access basis.

FIG. 36 depicts a flowchart for the access server 524 for access service triggering in an example of the invention. FIG. 36 begins in step 3600. In step 3602, the access server 524 waits for the next packet. The access server 524 then receives and processes a packet from the network device 512 in step 3604. In step 3606, the access server 524 checks if the protocol, sending address, and the destination address are known and allowed to pass. In one embodiment, the access server 524 checks the database system 522 for the determination made in step 3606. If the protocol, sending address, and the destination address are known and allowed to pass, the access server 524 checks if the sending address is consistent with the path in step 3608. If the sending address is not consistent with the path, the access server 524 registers a path security event in step 3610. The access server 524 then formats the packet with path information in step 3612 before returning to step 3602. If the sending address is consistent with the path, the access server 524 transmits the packet on the correct path with standard firewall access in step 3614 before returning to step 3602.

If the protocol, sending address, or the destination address are not known or are not allowed to pass, the access server 524 sets an external request timer in step 3616. The access server 524 then generates and transmits a request with the path information to the database system 522 in step 3618. The access server 524 then checks if a reply has been received or the external request timer has expired in step 3620. If the reply has not been received or the external request timer has not expired, the access server 524 checks if the reply is valid in step 3622. If the reply is invalid, the access server 524 discards the packet in step 3624 before returning to step 3602. If the reply is valid, the access server 524 then modifies the protocol and address filters based on the reply in step 3624 before returning to step 3614.

FIG. 37 depicts a flowchart for the database system 522 for access service triggering in an example of the invention.

FIG. 37 begins in step 3700. In step 3702, the database system 522 waits for the next request. The database system 522 then receives and processes the request in step 3704. In step 3706, the database system 522 then checks if the access server requester is known in step 3706. If the access server requester is not known, the database system 522 registers an unknown requester event in step 3708. The database system 522 then appends the reply to the packet in step 3710. The database system 522 then generates and transmits a decline reply to the access server 524 in step 3712 before returning to step 3702.

If the access server requester is known, the database system 522 then checks if the request is a filter appeal in step 3714. If the request is not a filter appeal, the database system 522 then checks if the request is a security event in step 3716. If the request is not a security event, the database system 522 registers an unknown action event in step 3717 before returning to step 3712. If the request is a security event, the database system 522 increments a path/device security record, appends the requester information to the request packet, and logs the event in step 3718. The database system 522 then checks if the request is a security event in step 3719. If the request is not a security event, the database system 522 then returns to step 3712. If the request is a security event, the database system 522 proceeds to step 3728.

If the request is a filter appeal, the database system 522 then identifies the user and network device 512 in step 3720. The database system 522 then checks if the access is valid in step 3722. If the access is invalid, the database system 522 returns to step 3718. If the access is valid, the database system 522 retrieves the user and network device profiles in step 3724. The database system 522 then generates access logic and appends the access logic to a reply in step 3726. The database system 522 then transmits the approve reply to the access server 524 in step 3728 before returning to step 3702.

Personal URL

The Internet currently uses Uniform Resource Locator (URL) addresses such as www.yahoo.com so that the address is in more human readable form than the network address. Unfortunately, user cannot distinguish themselves by a network address because the user's network address changes when the user's access point changes. FIGS. 5, 38 and 39 disclose one embodiment for a personal URL. In this embodiment, a network user may publish their location on the network by a user alias. A user alias is any alias that relates to the network address of a user. Logically linking a user's current network address with a user alias allows a user to be located wherever the user accesses the network.

FIG. 38 depicts a flowchart for a personal URL lookup in an example of the invention. FIG. 38 begins in step 3800. In step 3802, the database system 522 waits for the next packet. The database system 522 then receives and processes the packet including the user alias. The database system 522 then checks if a user alias translation is cached for this user in step 3806. If the user alias translation is cached, the database system 522 replies with the current network address of the user in step 3808 before returning to step 3802.

If the user alias translation is not cached, the database system 522 sets an external request timer in step 3810. The database system 522 then generates and transmits a request for user alias translation logic in step 3812. In one embodiment, the database system 522 transmits the request for user alias translation logic to the database system that contains the user access profile. The database system 522

then checks whether a reply was received or the external request timer expired in step 3814. If the reply was not received and the external request timer did not expire, the database system 522 returns to step 3814. If a reply was received or the external request timer did expire, the database system 522 checks if the reply was valid in step 3816. If the reply was invalid, the database system 522 replies with an invalid name message in step 3824 before returning to step 3802.

If the reply was valid, the database system 522 check if there is a current network address for the user in step 3818. If there is no current network address for the user, the database system 522 replies with user alias not currently in network message in step 3820 before returning to step 3802. If there is a current network address for the user, the database system 522 caches the user alias translation in step 3822 before returning to step 3808.

FIG. 39 depicts a flowchart for a personal URL update in an example of the invention. FIG. 39 begins in step 3900. In step 3902, the database system 522 wait for a request. The database system 522 then receives and processes the request to update the user alias in step 3904. In one embodiment, the database system receives the request from a database system that contains the user access profile. In another embodiment, the request comes from the access server 524. The database system 522 then checks if the requester is known in step 3906. If the requester is not known, the database system 522 registers an unknown requester event in step 3910. The database system 522 then transmits a reply with decline in step 3912 before returning to step 3902.

If the requester is known, the database system 522 then checks if the requester is allowed to update the user alias in step 3914. If the requester is not allowed, the database system 522 registers an unauthorized requester event in step 3916 before returning to step 3912. If the requester is allowed to update, the database system 522 then identifies and authenticates the user and current network address in step 3918. The database system 522 then checks if the access is valid in step 3920. If the access is invalid, the database system 522 increments the path/device security record in step 3922. The database system 522 also appends the requester information to the request and logs the event before returning to step 3912.

If the access is valid, the database system 522 then updates the user alias translation with the current network address in step 3924. The database system 522 then replies with an approve in step 3926 before returning to step 3902. In one embodiment of the invention, the database system 522 uses the personal DNS system 579 to perform the operations disclosed in FIGS. 38 and 39.

Access Network Caching—FIGS. 5, 40, 41, 42, 43, and 44

Predictive Caching

Users that are dialed into the network at a rate of 56 kbps or greater typically retrieve information at a rate of 2–4 kbps. Various equipment between the network access provider and the service provider cause this lower rate of transmission. Some examples of the equipment are network access provider equipment, service provider equipment, network backbone overloading, traffic shaping device, firewalls, and routers. One solution for compensating for the lower rate of transmission is caching. Computers typically contain a memory cache for specific application or devices. Firewalls and routers may also contain caches for data. Unfortunately, none of these caches are user specific.

FIGS. 5, 40, 41 and 42 disclose one embodiment of the invention for predictive end user caching. Predictive end

user caching advantageously improves user noticeable delays and slowdowns due to the lower transmission rate. The extension of data requests from the user network 510 beyond the access server 524 is eliminated when the data requested is cached in the database system. Also, using a predictive algorithm reduces delay by improving caching efficiency based on a user's predictable pattern.

FIG. 40 depicts a flowchart for the access server 524 for auditing in an example of the invention. FIG. 40 begins at step 4000. In step 4002, the user network 510 establishes a connection to the access server 524. In step 4004, the access server 524 generates and transmits a login request message to the database system 522. The access server 524 then checks if the database system allowed an access session to be established for the user in step 4006.

If the access session is not established, the access server 524 terminates in step 4020. If the access session is established, the access server 524 then checks if an access session tear down is requested in step 4012. If the access session tear down is requested, the access server 524 generates and transmits a tear down message with user and path information to the database system 522 in step 4014 before terminating in step 4020.

While no session tear down request is received, the access server 524 exchanges packets with the user network 510 depending on the service provided in step 4016. The access server 524 transmits all new packet destinations including the user and path information to the database system 522 in step 4018 before returning to step 4012. In one embodiment, the database system 522 stores the connect request message, the session reject message, the session accept message, the tear down message including the user and path information, and the packets including the user and path information in an audit database system 526. In one embodiment, the connect request message, the session reject message, the session accept message, the tear down message including the user and path information, and the packets including the user and path information are in the form of an audit message. An audit message is any message, information, or signaling that is audited while a user is accessing an access network 520.

FIG. 41 depicts a flowchart for the database system 522 for auditing in user predictive caching in an example of the invention. FIG. 41 begins in step 4100. In step 4102, the database system 522 waits for an audit message. In step 4104, the database system 522 receives and processes the audit message. The database system 522 then checks if the audit message is a session accept message or a tear down message in step 4106. If the audit message is not a session accept message or a tear down message, the database system 522 retrieves the session state object and updates the object with the audit message in step 4112 before returning to step 4102.

If the audit message is a session accept message or a tear down message, the database system 522 then checks if the audit message is a session accept message in step 4108. If the audit message is a session accept message, the database system 522 generates a session state object including the user, device, path, and session ID in step 4114 before returning to step 1802. If the audit message is a tear down message, the database system 522 retrieves the session state object and updates the object in step 4112. Also, the database system 522 stores the event in step 4116. In one embodiment, the database system 522 stores the event in the cache database system 574. The database system 522 removes the active reference from the session state object before returning to step 4102. In one embodiment of the invention, the database system 522 uses the audit database system 573 to perform the operations disclosed in FIG. 41.

FIG. 42 depicts a flowchart for the database system 522 for caching in user predictive caching in an example of the invention. FIG. 42 begins in step 4200. In step 4202, the database system 522 waits for the next event. The database system 522 processes the event in step 4204. In this embodiment, a request for data is a cache request. The database system 522 then checks if the event is a cache request in step 4206.

If the event is a cache request, the database system 522 identifies the user and retrieves the cached user data in step 4208. The database system 522 then checks if the requested data from the cache request is in the cached user data in step 4212. If the requested data is in the cached user data, the database system 522 replies with the requested data in step 4210 before returning to step 4202. If the requested data is not in the cached user data, the database system 522 retrieves the requested data from the service provider via the access server 524 in step 4214. The database system 524 then replies to the user network 510 with the requested data. Alternatively, in another embodiment, the access server 524 transfers the requested data from the service provider to the user network 510 and the database system 522.

In step 4222, the database system 522 updates the user access profile and the caching algorithm. The database system 522 then determines whether there is a change in the cached data based on the caching algorithm in step 4230. If there is no change in the cached data, the database system 522 returns to step 4202. If there is a change in the cached data, the database system 522 sets any new caching timers for the cached data in step 4236. The database system 522 also generates and transmits a translation change to the access server 524 to change the requested destination address to the database system 522 in step 4238 before returning to step 4202.

If the event is not a cache request, the database system 522 then checks if the event is an audit or timer request in step 4216. If the event is an audit or timer request, the database system 522 identifies the user and retrieves the user access profile in step 4218. The database system 522 then checks if the event is an audit request in step 4220. If the event is an audit request, the database system 522 proceeds to step 4222. If the event is not an audit request, the database system 522 refreshes the cached data from the service provider of the cached data based on the caching algorithm in step 4228. The database system 522 then resets the caching timer for the cached data in step 4234 before returning to step 4202.

If the event is not an audit or timer request, the database system 522 checks if the event is a reset event in step 4224. If the event is not a reset event, the database system 522 registers a unknown event received in step 4232 before returning to step 4202. If the event is a reset event, the database system 522 clears the user access profile statistics, caching timers, the user's cached data. In one embodiment, the user's cached data is stored in the cached database system 574 and the database system 522 uses the cached database system 574 to perform the operations disclosed in FIG. 42.

User Controlled Caching

Another solution to reduce user noticeable delays and slowdowns due to the lower transmission rate is user controlled caching. FIGS. 5, 40, 43 and 44 disclose one embodiment of the invention for user controlled caching. A user controls caching by creating a script of commands to cache data prior to the user accessing the network. For example, a user selects financial or local weather forecasts to cache, so when the user logs on to the network the information will be immediately available from the cache. Scripting has typi-

cally been done on general purpose computers. In a general purpose computer, the user sets up a sequenced set of commands to be executed when the script is executed. Unfortunately, this type of scripting has not been performed on a network cache.

In this embodiment of user controlled caching, the operation of the access server 524 is as described in FIG. 40. FIG. 43 depicts a flowchart for the database system 522 for auditing in user controlled caching in an example of the invention. FIG. 43 begins in step 4300. In step 4302, the database system 522 waits for an audit message. In step 4302, the database system 522 receives and processes the audit message. The database system 522 then checks if the audit message is a session accept message or a tear down message in step 4306. If the audit message is not a session accept message or a tear down message, the database system 522 retrieves the session state object and updates the object with the audit message in the cache in step 4312 before returning to step 4302.

If the audit message is a session accept message or a tear down message, the database system 522 then checks if the audit message is a session accept message in step 4308. If the audit message is a session accept message, the database system 522 generates a session state object in the cache including the user, device, path, and session ID in step 4314. The database system 522 then checks if the user has a pre-cache script in step 4318. If the user does not have a pre-cache script, the database system 522 returns to step 4302. If the user has a pre-cache script, the database system 522 generates and transmits a precache instruction set to the access server 524 to change the translation filter to access the database system 522 for destinations in the pre-cache script so that any requests for those destinations are fulfilled from the cache in step 4320. The database system 522 then sets cache refresh timers in step 4322 and returns to step 4302.

If the audit message is a tear down message, the database system 522 retrieves the session state object and updates the object in step 4316. Also, the database system 522 stores the event in step 4316. In one embodiment, the database system 522 stores the session state object and the state in the cache database system 527. The database system 522 removes the active reference from the session state object before returning to step 4302. In one embodiment of the invention, the database system 522 uses the audit database system 573 to perform the operations disclosed in FIG. 43.

FIG. 44 depicts a flowchart for the database system 522 for caching in user controlled caching in an example of the invention. FIG. 44 begins in step 4400. In step 4402, the database system 522 waits for the next event. The database system 522 processes the event in step 4404. In this embodiment, a request for data is a cache request. The database system 522 then checks if the event is a cache request in step 4406.

If the event is a cache request, the database system 522 identifies the user and retrieves the cached user data in step 4408. The database system 522 then checks if the requested data from the cache request is in the cached user data in step 4412. If the requested data is in the cached user data, the database system 522 replies with the requested data in step 4410 before returning to step 4402. If the requested data is not in the cached user data, the database system 522 retrieves the requested data from the service provider via the access server 524 in step 4414. The database system 522 then replies to the user network 510 with the requested data. Alternatively, in another embodiment, the access server 524 transfers the requested data from the service provider to the user network 510 and the database system 522.

In step 4422, the database system 522 updates the user access profile and the caching algorithm. The database system 522 then determines whether there is a change in the cached data based on the caching algorithm in step 4430. If there is no change in the cached data, the database system 522 returns to step 4402. If there is a change in the cached data, the database system 522 sets the new caching timer for the cached data in step 4436. The database system 522 also generates and transmits a translation change to the access server 524 to change the requested destination address to the database system 522 in step 4440 before returning to step 4402.

If the event is not a cache request, the database system 522 then checks if the event is an audit or timer request in step 4416. If the event is an audit or timer request, the database system 522 identifies the user and retrieves the user access profile in step 4418. The database system 522 then checks if the event is an audit request in step 4420. If the event is an audit request, the database system 522 proceeds to step 4422.

If the event is not an audit request, the database system 522 checks if the event is a script timer in step 4428. If the event is not a script timer, the database system 522 refreshes the cached data from the service provider of the cached data based on the caching algorithm in step 4438 before returning to step 4402. If the event is a script timer, the database system 522 executes the user commands or command set specified by the script timer event in step 4434. The database system 522 then resets the script timer for the cached data in step 4442 before returning to step 4402.

If the event is not an audit or timer request, the database system 522 checks if the event is a reset event in step 4424. If the event is not a reset event, the database system 522 registers a unknown event received in step 4432 before returning to step 4402. If the event is a reset event, the database system 522 clears the user access profile statistics, caching timers, the user's cached data. In one embodiment, the user's cached data is stored in the cached database system 574 and the database system 522 uses the cached database system 574 to perform the operations disclosed in FIG. 44.

Access Network Switching— FIGS. 5, 45, 46, and 47

Switching Access by a User

Access between users and service providers vary based on quality and security, which in turn determine the costs of the access. Users typically have a need to switch between types of access depending on the service offered or what stage in the service the user is in. For example, a user browses the amazon.com website for books using a standard Internet access. When purchasing the books, the user needs a more secure Internet access to ensure that the user's credit card number is not stolen by a hacker.

One prior solution for enhanced security is data Virtual Private Networks (VPN). Data VPN's are relatively static constructions which allow the extension of a user network to another location by extending the network over leased lines or shared Internet/Intranet facilities. However, VPN's require service anticipation, planning and expense beyond what the typical Internet and Intranet users possess. VPN's also do not provide dynamic switching between accesses. Another prior solution is Local Area Network emulation (LANE). LANE utilizes ATM transports to extend the reach of the user network. However, most Internet and Intranet users do not possess ATM equipment and expertise, and the backbone network is still IP.

FIGS. 5 and 45 disclose one embodiment for switching access by a user in an example of the invention. A user

selects a different access to a service provider and is switched to the access without the user's connection to their access provider being interrupted. FIG. 45 depicts a flowchart for switching access by a user in an example of the invention. In this embodiment, a user using standard Internet access selects a premiere Internet access during the setup of a service application. There are numerous variations in switching between access paths such as from premiere Internet access path to a lower quality Internet access path, but only one embodiment is shown for the sake of simplicity.

FIG. 45 begins in step 4500. In step 4502, a user through the network device 512 transmits a request using a standard Internet access path through the service network 530 for a service application residing in the network device 562. The service application in the network device 562 receives the request and transmits a query for the quality of service (QOS) Internet access desired by the requester to the network device 512 in step 4504. The service application in the network device 562 then receives and processes a request to switch access to check if the user selected a premium QOS Internet access in step 4516. If the user did not select a premium QOS Internet access, the user and service application exchange packets using the standard Internet access via the network device 512, the access server 524, the service network 530, the access server 554, and the network device 562 in step 4516. The service application in the network device 562 returns the control to the user in step 4514 to select another service application in step 4502.

If the user selects a premium QOS Internet access, the service application in the network device 562 generates and transmits an authentication and authorization instruction for the premium QOS Internet access to the database system 522 to check if the switch of access is allowed. The database system 522 receives and processes the authentication and authorization instruction. The database system 522 then generates and transmits a premium access instruction to establish premium Internet access between the access server 524 and the access server 554 via the service network 540. In one embodiment, the database system 522 transmits the premium access instruction to the service network 540 to establish premium Internet access between the access server 524 and the access server 554. The database system 522 then generates and transmits the premium access instruction to the access server 524 to connect the network device 512 to the service network 540 for the premium Internet access. The database system 522 also generates and transmits the premium access instruction to the access server 554 to connect the network device 562 to the service network 540 for the premium Internet access.

Once the premium Internet access is established, the user and service application exchange packets using the premium Internet access via the network device 512, the access server 524, the service network 540, the access server 554, and the network device 562 in step 4512. The database system 522 then generates and transmits a disconnect instruction to the access server 524, the service network 540, and the access server 554 to tear down the premium Internet access in step 4518. The database system 522 then generates and transmits usage and performance statistics for the service application in the network device 562 in step 4520 before returning to step 4514. In one embodiment, the database system 522 uses a cross connect system 583 to perform the operations disclosed in FIG. 45.

Switching Access by a Service Provider

FIGS. 5 and 46 disclose one embodiment for switching access by a service provider in an example of the invention. The service provider selects a different access to a user and

41

is switched to the access without the user or service provider's connection to their access provider being interrupted. FIG. 46 depicts a flowchart for switching access by a service provider in an example of the invention. In this embodiment, the service provider selects a toll free premiere Internet service for a user using standard Internet access. The service provider pays for the changes in the toll free premium Internet service instead of the user—similar to toll free phone calls.

FIG. 46 begins in step 4600. In step 4602, the service application in the network device 562 waits for a request for a service. The user through the network device 512 transmits a request using a standard Internet access through the service network 530 for the service application in the network device 562 in step 4604. The service application in the network device 562 receives and processes the request. The service application then selects a toll free premiere quality of service (QOS) Internet access to the network device 512 in step 4606. The service application in the network device 562 generates and transmits an authentication instruction to the database system 522 in step 4608. The database system 522 receives and processes the authentication instruction. In step 4612, the database system 522 then generates and transmits a premium access instruction to establish the toll free premium Internet access between the access server 524 and the access server 554 via the service network 540.

Once the toll free premium Internet access is established, the user and service application exchange packets using the premium Internet access via the network device 512, the access server 524, the service network 540, the access server 554, and the network device 562 in step 4612. Once the service is completed, the database system 522 then generates and transmits a disconnect instruction to the access server 524, the service network 540, and the access server 554 to tear down the toll free premium Internet access in step 4614. The database system 522 then generates and transmits usage and performance statistics for the service application in the network device 562 in step 4616. Once the service is completed, the service application in the network device 562 returns the control to the user in step 4618 to select another service application in step 4602. In one embodiment, the database system 522 uses the cross connection system 583 to perform the operations disclosed in FIG. 46. In another embodiment, a third party such as the government performs the switching for surveillance or monitoring purposes.

Dynamic Switching Access

FIGS. 5 and 47 disclose one embodiment for dynamic switching access in an example of the invention. In a prior solution, a single access link from the user network to the access server is linked to a dedicated network in a single access session. However, users need to switch between different networks such as Internet and Intranets without tearing down the existing access session. No other systems allow the access server to be controlled by the data stream the access server is passing. Analog modems monitor the character stream for an escape sequence. The escape sequence notifies the modem that the data is for modem control and not for application data. However, no prior systems have implemented this functionality for a packet data network.

FIG. 47 depicts a flowchart for dynamic switching access in an example of the invention. In this embodiment, a user during an access session may switch networks without having the access session torn down. The access server 524 receives a control instruction from the user to switch from one dedicated service network 530 to another dedicated service network 540. FIG. 47 begins in step 4700. In step

42

4702, the access server 524 waits for a packet. In this embodiment, a request is in the format of a packet. The access server 524 then receives and processes the packet from the network device 512 in step 4704. The access server 524 checks if the packet is encoded for access control in step 4706. If the packet is not encoded for access control, the access server 524 processes the packet using standard access logic in step 4714 before returning to step 4702.

If the packet is encoded for access control, the access server 524 identifies the physical access path characteristics in step 4708. The access server 524 then checks if the access control is allowed in step 4710. If access control is not allowed, the access server 524 registers and logs an illegal access event in step 4716. The access server 524 then discards the packet in step 4718 before returning to step 4702.

If access control is allowed, the access server 524 generates and transmits an access control instruction to the database system 522 in step 4712. The database system 522 then receives and processes the access control instruction. The database system 522 identifies, authenticates, and authorizes the user and the requesting access server using the packet and path in step 4720. The database system 522 then retrieves the user access profile and the network device profile in step 4722. The database system 522 then checks if access control is allowed for the user and the network device based on the user access profile and the network device profile in step 4724. If access is not allowed, the database system 522 proceeds to step 4716.

If access is allowed, the database system 522 generates and transmits an access instruction to the access server 524 to update the access logic to switch to the service network 540 in step 4726. The database system 522 logs the access change in step 4727. In one embodiment, the database system 522 logs the access change in the audit system 572. The database system 522 also generates and transmits a reply with the new access status to the network device 512 in step 4728 before returning to step 4702. In one embodiment, the database system 522 uses the cross connect system 583 to perform the operations disclosed in FIG. 47.

Access Network Destination Control—FIGS. 5, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, and 59

Network Failover

Network devices become unavailable for a various reasons such as busy, failure, or overload. For network failover, when a destination network device is unavailable, network device requests are manually re-routed to a secondary network device. In order to avoid this manual rerouting, one prior art solution is the sharing of a network address between multiple processors in a box or multiple machines in a cluster. However, the requirement of destination and secondary devices residing in the same box or cluster makes this solution unacceptable for large systems.

Another prior art system is Network Address Translation, which translates a private network address to a public network address for a period of time. This system does not provide the capability of a network address to represent multiple failover destinations. Another solution is the Common Object Request Broker Architecture (CORBA), which provides the user a list of potential "logical" network devices to post the network device request. However, the user must implement CORBA interfaces and interactively select back up processes. The transaction context is lost when the primary fails. The user must re-establish the context. Unfortunately, none of these solutions provide an automatic translation from an unavailable destination network device to a secondary network device without operator

or user intervention, where the destination and secondary devices are not within the same box or cluster.

FIGS. 5, 48 and 49 disclose one embodiment for handling network address failover in an example of the invention. In a failover scenario, a destination network device fails or overloads, which makes the destination network device unavailable, and a secondary network device replicates the destination network device to give the user a fault tolerant appearance of the destination network device. Requests for the destination network device are translated to the secondary network device.

FIG. 48 depicts a flowchart for the access server 554 for network address failover in an example of the invention. FIG. 48 begins in step 4800. In step 4802, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 4804. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 4806, the access server 554 detects that the destination network device 562 fails and checks if the destination network address is enabled for failover in step 4806. If the destination network address is not enabled for failover, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 4812 before returning to step 4802.

If the destination network address is enabled for failover, the access server 554 checks if an address translation is active for the destination network device 562 in step 4808. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 4812 before returning to step 4802. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for failover in step 4810. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 4812 before returning to step 4802.

FIG. 49 depicts a flowchart for the database system 522 for network address failover in an example of the invention. FIG. 49 begins in step 4900. In step 4902, the database system 522 waits for a state event message or a timer event from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or a network device timer expires in step 4904. The database system 522 checks if the network device timer expired in step 4906.

If the network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout in step 4908. The database system 522 then retrieves and updates and the secondary network device's 564 profile to indicate that the destination network device 562 is unavailable in step 4910. In step 4912, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer has not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 4914. In step 4916, the database system 522 checks if the destination network device 562 is known. If the destination network device 562 is not known, the database system 522 generates

and transmits an unknown resource event before returning to step 4902. If the destination network device 562 is known, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics in step 4918. The database system 522 also resets the network device timer. The database system 522 then checks if the state event message is a failure or start event in step 4920.

If the state event message is not a failure or start event, the database system 522 checks if the load event threshold for the destination network device 562 is reached in step 4930. If the load event threshold is not reached, the database system 522 returns to step 4902. If the load event threshold is reached, the database system 522 proceeds to step 4926.

If the state event message is a failure or start event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 4922. In step 4924, the database system 522 checks if the state event message is a failure event. If the state event message is not a failure event, the database system 522 generates new active device profiles and then returns to step 4902. If the state event message is a failure event, the database system 522 proceeds to step 4926.

In step 4926, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address in step 4932 before returning to step 4902. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event in step 4928 before returning to step 4902. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 49.

Network Busy Forwarding

In a Public Switched Telephone Network (PSTN), a busy signal is sent to the caller when the called number destination is busy and thus unavailable. The caller can then select alternate actions such as delaying, retrying, or re-routing the request. Unfortunately, the PSTN handling of busy calls has not been applied to network devices in a data network. FIGS. 5, 50 and 51 disclose one embodiment for network busy forwarding in an example of the invention. When a destination network device 562 is busy, an access server 554 forwards the packets to a secondary network device 564. Requests for the destination network device are translated to the secondary network device.

FIG. 50 depicts a flowchart for the access server 554 for network busy forwarding in an example of the invention. FIG. 50 begins in step 5000. In step 5002, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5004. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5006, the access server 554 checks if the destination network device 562 is busy. If the destination network device 562 is not busy, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5012 before returning to step 5002.

If the destination network device 562 is busy, the access server 554 checks if an address translation is active for the destination network device 562 in step 5008. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step

5012 before returning to step 5002. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for failover in step 5010. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 5012 before returning to step 5002.

FIG. 51 depicts a flowchart for the database system 522 for network busy forwarding in an example of the invention. FIG. 51 begins in step 5100. In step 5102, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or the network device timer expired in step 5104. The database system 522 checks if a network device timer expired in step 5106.

If a network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout such as the network device being busy or unavailable in step 5108. The database system 522 then retrieves and updates and the secondary network device's 564 profile to indicate that the destination network device 562 is busy or unavailable in step 5110. The database system 522 also resets the network device timer. In step 5112, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer is not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 5114. In step 5116, the database system 522 checks if the destination network device 562 is known. If the destination network device 562 is not known, the database system 522 generates and transmits an unknown resource event before returning to step 5102. If the destination network device 562 is known, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics in step 5118. The database system 522 then checks if the state event message is a failure/busy or start/available event in step 5120.

If the state event message is not a failure/busy or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5130. If the load event threshold is not reached, the database system 522 returns to step 5102. If the load event threshold is reached, the database system 522 proceeds to step 5126.

If the state event message is a failure/busy or start/available event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 5122. In step 5124, the database system 522 checks if the state event message is a failure/busy event. If the state event message is not a failure/busy event, the database system 522 generates an address translation clear request in step 5132 before returning to step 5102. The address translation clear request adds a new active device profile or clears the existing translation. If the state event message is a failure/busy event, the database system 522 proceeds to step 5126.

In step 5126, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address if a busy

translation exists in the destination network device's 562 profile in step 5134 before returning to step 5102. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5128 before returning to step 5102. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 51.

Busy Flag

FIGS. 5 and 52 disclose one embodiment for a busy flag when the destination network device is busy in an example of the invention. When a destination network device 562 is busy and thus unavailable, an access server 554 replies to the user with a busy flag. FIG. 52 depicts a flowchart for the access server 554 for a busy flag when the destination network device is busy in an example of the invention. In one embodiment, the busy flag is a busy screen in HyperText Markup Language. FIG. 52 begins in step 5200. In step 5204, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5204. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5206, the access server 554 checks if the destination network device 562 is busy. If the destination network device 562 is not busy, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5210 before returning to step 5202. If the destination network device 562 is busy, the access server 554 replies to the user with a busy flag in step 5208 before returning to step 5202.

Timeout

FIGS. 5 and 53 disclose one embodiment for forwarding if the destination network device times out in an example of the invention. In this embodiment, a destination network device 562 receives a request packet from a user but fails to reply to the user within a pre-determined time. The failure to reply within a pre-determined time indicates the destination network device 562 is unavailable. The access server 554 then redirects the packet to a secondary network device 564 to handle forwarding when the destination network device fails to respond with the pre-determined time.

FIG. 53 depicts a flowchart for the access server 554 for forwarding if the destination network device times out in an example of the invention. FIG. 53 begins in step 5300. In step 5302, the access server 554 waits for the next packet or event. The access server 554 then checks whether a reply timeout event is received in step 5304. If a reply timeout event is not received, the access server 554 then receives and processes the packet in step 5306. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5308, the access server 554 checks if an address translation is available for the destination network device 562. If an address translation is not available for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5318 before returning to step 5302. If an address translation is available for the destination network device 562, the access server 554 sets a reply timer and caches the packet for the secondary network device 564. The access server 554 then transmits the packet with standard access logic to the destination network device 564 in step 5318 before returning to step 5302.

In step 5312, the access server 554 receives the reply timeout event and retrieves the cached request. The access server 554 then processes the reply timeout event and the cached request in step 5314. In step 5315, the access server

554 then replaces the packet's destination network address with the cached secondary network address. The access server 554 then transmits the packet to the secondary network device 564 with the standard access logic in step 5318 before returning to step 5302. In one embodiment, upon reply, the access server 554 deletes the cached packet copy.

Scheduled Alias Resolution

Network devices become unavailable for various reasons such as busy, failure, or overload. Also, service provider need to utilize network resources to improve efficiency and avoid network device latency. In voice telephony signaling networks, an SS7 SCP or an automatic call distributor (ACD) distributes calls to end user call centers. The SCP receives a 800 TCAP query and performs a database lookup to translate the 800 telephone number into a destination telephone number. The SCP may use the requester information and the time of day to perform the database lookup. Some problems with this system are SS7 signaling must be used and the system is limited by voice network constraints. Also, the phone numbers only allow numeric numbers for aliases. In data networks, prior art known as Object Request Brokers (ORB) in a Common Object Request Broker Architecture (CORBA) advertise process resources to a requester. The ORB also registers multiple instances of the same type of process resource.

FIGS. 5, 54, and 55 disclose one embodiment for scheduled alias resolution in an example of the invention. In this embodiment, an alias, domain name/local name, network address, or macro for a network resource is translated to another alias for another network resource based on a configurable schedule. Advantageously, load balancing of network devices is improved by scheduling different alias resolutions for different times/conditions. One problem with CORBA is the requester and resources must implement CORBA interfaces. Scheduled alias resolution is compatible with existing data network DNS implementations. Also, the load balancing, security, and failover may be based on the user access profile and the network device state.

FIG. 54 depicts a flowchart for the access server 554 for schedule alias resolution in an example of the invention. FIG. 54 begins in step 5400. In step 5402, the access server 554 waits for the next request. The access server 554 then receives and processes a request from the network device 512 in step 5404. In this embodiment, the information including an alias is in the form of a request. In step 5406, the access server 554 checks if a current translation for the alias exists. If a current translation for the alias exists, the access server 554 translates the alias to the appropriate network address for a network device in user network 560 in step 5408. The access server 554 then replies to the network device 512 in step 5410 before returning to step 5402.

If a current translation for the alias does not exist, the access server 554 checks if a known access profile exists in the database system 522 exists for this user in step 5412. If a known access profile does not exist in the database system 522 for the user, the access server 554 generates and transmits an alias resolution failure message to the network device 512 in step 5414 before returning to step 5402. If a known access profile does exist in the database system 522 for the user, the access server 554 sets an external request timer in step 5416. The access server 554 then generates and transmits an alias translation request to the database system 522 in step 5418. The access server 554 then checks if a reply is received or the external request timer is expired in step 5420. If the reply is not received or the external request timer is not expired, the access server 554 returns to step

5420. If the reply is received or the external request timer is expired, the access server 554 checks if the reply was valid in step 5422. If the reply is invalid, the access server 554 returns to step 5414. If the reply is valid, the access server 554 then caches the alias translation for this user with time to live parameters in step 5424 before returning to step 5408.

FIG. 55 depicts a flow chart for the database system 522 for scheduled alias resolution in an example of the invention. FIG. 55 begins in step 5500. In step 5502, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 554 in step 5504. In step 5506, the database system 522 then checks if the access server requester is known in step 5506. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 5508. The database system 522 appends the reply to the packet and transmits the reply to the access server 554 in step 5510. The database system 522 replies with a decline message to the access server 554 in step 5512 before returning to step 5502.

If the access server requester is known, the database system 522 then checks whether the request is a translation appeal in step 5514. If the request is not a translation appeal, the database system 522 checks if the request is a security event in step 5516. If the request is a security event, the database system 522 increments a path/device security record in step 5522 before returning to step 5512. If the request is not a security event, the database system 522 registers an unknown event in step 5530 before returning to step 5502. If the request is a translation appeal, the database system 522 identifies and authenticates the user and the network device 512 in step 5518. If the access is not valid, the database system 522 returns to step 5522. If the access is valid, the database system 522 retrieves the user and network device's 512 profiles in step 5524. The database system 522 then generates the translation logic and calculates the translation time to live in step 5526. The database system 522 then appends the translation logic and the translation time to a reply to the access server 554. The database system 522 then replies with an approve message to the access server 554 in step 5528.

Forwarding

In a PSTN, a person may forward calls from their original phone number to another phone number where the person may be reached. Unfortunately, the PSTN handling of call forwarding has not been applied to network devices in a data network. FIGS. 5, 56 and 57 disclose one embodiment for destination controlled forwarding in an example of the invention. An access server 554 forwards the packets for a destination network device 562 that is unavailable to a secondary network device 564. Requests for the destination network device 562 are translated to the secondary network device 564.

FIG. 56 depicts a flowchart for the access server 554 for destination controlled forwarding in an example of the invention. FIG. 56 begins in step 5600. In step 5602, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5604. In this embodiment, the access server 554 receives and processes information in the form of a packet. The access server 554 then checks if the packet is a reply in step 5606. If the packet is a reply, the access server 554 retrieves and deletes the cached destination network address in step 5608 before proceeding to step 5614.

If the packet is not a reply, the access server 554 checks if an address translation is active for the destination network

device 562 in step 5610. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5614 before returning to step 5602. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for destination controlled forwarding in step 5612. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 5614 before returning to step 5602.

FIG. 57 depicts a flowchart for the database system 522 for destination controlled forwarding in an example of the invention. FIG. 57 begins in step 5700. In step 5702, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or a network device timer expires in step 5704.

The database system 522 processes the state event message and authenticates the destination network device 562 in step 5706. In step 5708, the database system 522 checks if the access server requester is known. If the access server requester is not known, the database system 522 generates and transmits an unknown requester event before returning to step 5702. If the access server requester is known, the database system 522 retrieves and updates the access server's 554 profile with availability and performance statistics in step 5710. The database system 522 then checks if the state event message is a failure/forward or start/available event in step 5712.

If the state event message is not a failure/forward or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5716. If the load event threshold is not reached, the database system 522 returns to step 5702. If the load event threshold is reached, the database system 522 proceeds to step 5722.

If the state event message is a failure/forward or start/available event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 5718. In step 5720, the database system 522 checks if the state event message is a failure/forward event. If the state event message is not a failure/forward event, the database system 522 generates and transmits an address translation clear request message to the access server 554 in step 5726 before returning to step 5702. If the state event is a start event, then a new availability profile is generated. If the state event message is a failure/busy event, the database system 522 proceeds to step 5722.

In step 5722, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address if a busy translation exists in the destination network device's 562 profile in step 5728 before returning to step 5702. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5724 before returning to step 5702. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 57.

Network Endpoint Availability Management

FIGS. 5, 58, and 59 disclose one embodiment for network endpoint availability management in an example of the invention. The features of failover, busy, busy flag, timeout, and forwarding are combined into this embodiment. FIG. 58 depicts a flowchart for the access server 554 network endpoint availability management in an example of the invention. FIG. 58 begins in step 5800. In step 5802, the access server 554 waits for the next packet or event. The access server 554 then checks whether a reply timeout event is received in step 5804.

If a reply timeout event is not received, the access server 554 then receives a data packet and processes the packet in step 5806. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5808, the access server 554 checks if the packet is a reply. If the packet is a reply, the access server 554 clears the cached request and clears a reply timer in step 5810 before returning to step 5804. If the packet is not a reply, the access server 554 checks if an address translation is active for the destination network device 562 in step 5812. If an address translation is active for the destination network device 562, the access server 554 proceeds to step 5824. If an address translation is not active for the destination network device 562, the access server 554 checks if the destination address is busy. If the destination address is not busy, the access server 554 proceeds to step 5830. If the destination address is not busy, the access server 554 checks if a forward address translation is available in step 5814. If the forward address translation is not available, replies to the user with a busy flag in step 5818 before returning to step 5802. If the forward address translation is available, the access server 554 proceeds to step 5824.

If a reply timeout event is received, the access server 554 receives the reply timeout event and the cached request in step 5820. The access server 554 then processes the reply timeout event and the cached request in step 5822. In step 5824, the access server 554 then replaces the packet's destination network address with the cached secondary network address. The access server 554 then transmits the packet to the secondary network device 564 with the standard access logic in step 5824.

In step 5826, the access server 554 checks if the forward address is busy. If the forward address is busy, the access server 554 returns to step 5818. If the forward address is not busy, the access server 554 sets a reply timer and caches the request in step 5828. In step 5830, the access server 554 then transmits the packet with standard access logic before returning to step 5802.

FIG. 59 depicts a flowchart for the database system 522 for network endpoint availability management in an example of the invention. FIG. 59 begins in step 5900. In step 5902, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives the state event message from the destination network device 562 or the network device timer expired in step 5904. The database system 522 checks if the network device timer expired in step 5906.

If the network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout such as the network device being busy or unavailable in step 5908. The database system 522 then retrieves and updates the secondary network device's 564 profile to indicate that the destination network device 562 is busy or unavailable in step 5910. The database system 522 also resets the network device timer. In step 5912, the database system 522 generates and transmits an

51

address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer has not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 5914. In step 5916, the database system 522 checks if the access server requester is known. If the access server requester is not known, the database system 522 generates and transmits an unknown resource event before returning to step 5902. If the access server requester is known, the database system 522 retrieves and updates the access server's 554 profile with availability, activity logs, and performance statistics in step 5918. The database system 522 then checks if the state event message is a failure, busy, forward or start/available event in step 5920.

If the state event message is not a failure, busy, forward or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5930. If the load event threshold is not reached, the database system 522 returns to step 5102. If the load event threshold is reached, the database system 522 proceeds to step 5926.

If the state event message is a failure, busy, forward, or start/available event, the database system 522 retrieves and updates the destination network device 562 and secondary network device's 564 profile in step 5922. In step 5124, the database system 522 checks if the state event message is a failure, busy, or forward event. If the state event message is not a failure/busy event, the database system 522 generates and transmits an address translation clear request to the access server 554 if a busy translation exists in step 5934 before returning to step 5902. If the state event message is a failure, busy, or forward event, the database system 522 proceeds to step 5926.

In step 5926, the database system 522 checks if a secondary network device 564 is available for forwarding. If a secondary network device 564 is available, the database system 522 checks if the state event message is a failure or busy event in step 5936. If the state event message is a failure or busy event, the database system 522 generates and transmits a busy address translation request message to the access server 554 in step 5932 before returning to step 5902. If the state event message is not a failure or busy event, the database system 522 generates and transmits a forward address translation request message to the access server 554 to change packets' destination network address to the secondary network address in step 5938 before returning to step 5902. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5928 before returning to step 5902. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 59.

Access Network Audit Server—

FIGS. 5, 60, 61, 62, and 63

Service Capability Monitor

FIGS. 5, 60, 61, 62, and 63 disclose one embodiment for service capability monitor in an example of the invention. In a prior system, Java applet technology and Sun's JINI project delivers communication and state mechanisms from the function provider to the function requesters. Java applet technology allows a mechanism for device control without any coordinating control agent. JINI provides a registry which maintains capabilities like a CORBA ORB as a means for primitive control. One problem is that JINI focuses on

52

the distribution of mechanisms instead of the interworking of mechanisms in a standardized way. In this embodiment of service capability control, a network access provider monitors the validity and state of the application level request to a network service provider. The network access provider monitors performance of their network and the service provider's network service requiring an understanding of the other provider's service requested. Access providers such as Internet service providers could mitigate their liability and increase the service's performance by extending their service view to the user base. Specific service environment access points may be monitored and service access rights and restrictions may be enforced. This embodiment provides an interworking mechanism that provides a dynamic interface to a service based network. The monitoring interface must be dynamic because different service have different potential states, different legal and illegal state transitions, and different communication mechanisms.

FIG. 60 depicts a flowchart for a firewall 526 for service capability monitor in an example of the invention. FIG. 60 begins in step 6000. In step 6002, the firewall 526 waits for a packet. The firewall 526 then receives and processes a packet from the network device 512 in step 6004. In step 6006, the firewall 526 performs standard firewall functions. The firewall 526 then copies the packets and transmits the packet to the destination and the database system 522 in step 6008 before returning step 6002.

FIG. 61 depicts a service capability monitor software architecture for a service capability monitor in an example of the invention. A service capability monitor software architecture 6100 comprises a service broker tokenizer process B 6112, a service broker state map process C 6114, a network A controller process A 6116, a service A signal tokenizer process E 6122, a service A state map process F 6124, a service A controller process D 6126, a service B signal tokenizer process H 6132, a service B state map process F 6134, and a service B controller process D 6136. The network A controller process A 6116 includes a service broker session state objects 6118. The service A controller process D 6126 includes a service A session state objects 6128. The service B controller process D 6136 includes a service B session state object 6138.

The network A controller process A 6116 is connected to the service broker tokenizer process B 6112, the service broker state map process C 6114, the service A controller process D 6126, and the service B controller process D 6136. The service A controller process D 6126 is connected to the service A signal tokenizer process E 6122 and the service A state map process F 6124. The service B controller process D 6136 is connected to the service B signal tokenizer process H 6132 and the service B state map process F 6134.

FIG. 62 depicts a flowchart for the network logic for service capability monitor in an example of the invention. FIG. 62 begins in step 6200. In step 6202, the database system 522 waits for the packet. The database system 522 then receives and processes the packet from the firewall 526 in step 6204. The database system 522 then identifies and authorizes the requester in step 6206. The database system 522 checks if the requester is known in step 6208.

If the requester is known, the network A controller process 6116 writes the packet to process B—service broker signal tokenizer process 6112 in step 6210. The process B 6112 then replies with the function token and parameters to the process A 6116 in step 6212. The database system 522 then checks if a service session state object (SSO) 6118 exists for this session in step 6214. If the SSO 6118 does not exist, the database system 522 creates a SSO 6118 for this session and

53

initializes the current state in step 6216. If the SSO 6118 exists, the database system 522 then proceeds to step 6218. In step 6218, process A 6116 writes the state token and current state to process C—service broker state map process 6114 in step 6218. The process C 6114 then replies with a transition token and parameters to the process A 6116 in step 6220. The process A 6116 applies the transition token and the parameters to the SSO 6118 to update the current state and setting actions in step 6222.

In step 6224, the database system 522 checks if there is an action indicated by the SSO 6118. If there is not an action indicated by the SSO 6118, the database system 522 proceeds to step 6228. If there is an action indicated by the SSO 6118, the database system 522 performs the transition action specified by the SSO 6118 in step 6226. In step 6228, the database system 522 updates the session sliding usage accumulators. The database system 522 then checks if there is an action in the SSO 6118 for this usage in step 6230. If there is no action, the database system 522 returns to step 6202. If there is an action in the SSO 6118, the database system 522 performs the usage action specified in the SSO 6118 before returning to step 6202.

If the requester is not known, the database system 522 checks if there is a network A state object in step 6234. If there is a network state object, the database system 522 proceeds to step 6238. If there is no network state object, the database system 522 creates a network A state object and initializes the current state in step 6236. In step 6238, the database system 522 increments network A state object usage for this event. The database system 522 then checks if there is an action in the network A state object for this usage in step 6240. If there is no action in the network A state object for this usage, the database system 522 returns to step 6202. If there is an action in the network A state object for this usage, the database system 522 performs the usage action specified in the network A state object in step 6242 before returning to step 6242.

FIG. 63 depicts a flowchart for the service logic for service capability monitor in an example of the invention. FIG. 63 begins in step 6300. In step 6302, process D service A controller process 6126 waits for the packet. The process D 6126 then receives and processes the packet from the process A network A controller 6116 in step 6304. The process D 6126 then identifies and authorizes the requester in step 6306. The process D 6126 checks if the requester is known in step 6308.

If the requester is known, the process D 6126 writes the request to process E service A signal tokenizer process 6122 in step 6310. The process E 6122 then replies with the state token and parameters to the process D 6126 in step 6312. The database system 522 then checks if a service session state object (SSO) 6128 exists for this session in step 6314. If the SSO 6128 does not exist, the database system 522 creates a SSO 6128 for this session and initializes the current state in step 6316. If the SSO 6118 exists, the database system 522 then proceeds to step 6318. In step 6218, the process D 6126 writes the state token and current state to process F—service state map process 6124 in step 6318. The process F 6124 then replies with a transition token and parameters to the process D 6126 in step 6320. The process D 6126 applies the transition token and the parameters to the SSO 6128 to update the current state and setting actions in step 6322.

In step 6324, the database system 522 checks if there is an action indicated by the SSO 6128. If there is not an action indicated by the SSO 6128, the database system 522 proceeds to step 6328. If there is an action indicated by the SSO

54

6128, the database system 522 performs the transition action specified by the SSO 6128 in step 6326. In step 6328, the database system 522 updates the session sliding usage accumulators. The database system 522 then checks if there is an action in the SSO 6128 for this usage in step 6330. If there is no action, the database system 522 returns to step 6302. If there is an action in the SSO 6128, the database system 522 performs the usage action specified in the SSO 6128 before returning to step 6302.

If the requester is not known, the database system 522 checks if there is a service A state object in step 6334. If there is a service state object, the database system 522 proceeds to step 6338. If there is no service state object, the database system 522 creates a service A state object and initializes the current state in step 6336. In step 6338, the database system 522 increments service A state object usage for this event. The database system 522 then checks if there is an action in the service A state object for this usage in step 6340. If there is no action in the service A state object for this usage, the database system 522 returns to step 6302. If there is an action in the service A state object for this usage, the database system 522 performs the usage action specified in the service A state object in step 6342 before returning to step 6342.

EXAMPLE

The following is one example of the access communication system that integrates many of the features described in the various embodiments of the inventions above. In the example below, a subscriber called Bank has many employees. The Bank has access to the network architecture 500 through an access provider called AccProv1. In this example, a bank employee is retrieving a stock quote from the yahoo.com website and then switches to request a streaming video from the yahoo.com website. Yahoo's web-server is running in the user network 562. Yahoo has access to the network architecture 500 through an access provider called AccProv2.

A system administrator for AccProv1 configures and optionally starts an access execution environment for the subscriber Bank in the local database system 570. The system administrator also starts a subscriber proxy for the Bank that runs in the access execution environment for the Bank. The subscriber proxy includes transport handlers, configuration handlers, and security handlers. The AccProv1 system administrator starts access execution environments for the maximum number of concurrent users in the local database system 570. The system administrator also starts generic proxies for users in the local database system 570. These generic proxies become user proxies after the user access profile is retrieved for the user.

The system administrator for AccProv2 starts an access execution environment for a service for Yahoo in the local database system 590. The system administrator also starts a service proxy for Yahoo's service that runs in the access execution environment for Yahoo. A system administrator for the Bank then sets up the user access profiles for each employee by inheriting attributes for the user access profile from capability classes or groups that the user belongs to. The user access profile includes the network shell for the user.

The Bank employee then logs on to the access server 524 using a user ID. The Bank system administrator set up. Besides user ID, a user may login to the access server 524 using a prepaid account code, a bank card number, an access card account code, or a user ID including a delimiter for user

access mobility. Upon connection, the access server 524 creates a device session in an access execution environment for the user. Then in response to the user login, the database system 522 retrieves the user access profile for the Bank employee. For a prepaid account code, a bank card number, an access card account code, and a user ID with a delimiter, the database system 522 may retrieve the user access profile from a database system external to the local database system 570. Also, the database system 522 may retrieve the user access profile from a database system external to the local database system 570 for global authentication.

Once the user access profile is retrieved, the access server 524 creates a user session in the access execution environment for the user and binds the device session with the user session. The configuration handler of the user proxy includes the network shell retrieved with the user access profile. The access server 524 then generates and transmits a list of available services in a service based directory to the Bank employee based on the user access profile.

The Bank employee then selects a service to check a stock quote for IBM by transmitting a request for service to the access server 524. The request for service may be in the form of a selection from the service based directory or an alias for alias translation for the network shell using domain name scoping or inbound DNS lookup. The access server 524 processes and transfers the request for service to the user proxy for the Bank employee. The user proxy transfers the request to the service proxy for Yahoo. The service proxy processes and transfers the request for service to Yahoo's web server. The database system 522 translates the service request for Yahoo to a private fulfillment address for one of Yahoo's web servers. The access server 554 then determines if a private destination address is available. If the private destination address is unavailable, the access server 554 could perform many actions including forwarding the request to another Yahoo web server or replying with a busy flag. The access server 524 also binds the user session and the device session with the selected service session by exchanging reference ID's.

The Bank employee then requests another stock quote for Oracle. The access server 524 then determines this stock quote was already cached based on the user's predictable pattern of requesting quotes for this company or a group of software companies. These prior requests for stock quotes of software companies by the Bank employee were audited to derive the Bank employee's predictable pattern. Alternatively, the request for the stock quote for Oracle could have been setup by the user in a script of commands to cache the quote in advance.

The Bank employee then requests a streaming video from Yahoo. The access server 524 switches the access for a premiere access based on the request for streaming video from the Bank employee. Alternatively, Yahoo initiates a switch of access for toll-free Internet service to the Bank employee who is a preferential customer. In order to be able to run the streaming video, the database system 522 also inherits user profile information from a class for streaming video user and updates the user access profile with the user profile information. The user proxy transfers the request for streaming video to the subscriber proxy. The subscriber proxy transfers the request to the service proxy. The service proxy then transfers the request to the Yahoo for one of their streaming video servers. Yahoo's streaming video server then provides the streaming video to the Bank employee.

Conclusion

The access network operates as described in response to instructions that are stored on storage media. The instruc-

tions are retrieved and executed by a processor in the access network. Typically, the processor resides in a server or database. Some examples of instructions are software, program code, and firmware. Some examples of storage media are memory devices, tape, disks, integrated circuits, and servers. The instructions are operational when executed by the processor to direct the network to operate in accord with the invention. The term "processor" refers to a single processing device or a group of inter-operational processing devices. Some examples of processors are computers, integrated circuits, and logic circuitry. Those skilled in the art are familiar with instructions, processors, and storage media.

Those skilled in the art will appreciate variations of the above-described embodiments that fall within the scope of the invention. As a result, the invention is not limited to the specific examples and illustrations discussed above, but only by the following claims and their equivalents.

1 claim:

1. A method of operating an access system including an access server to provide access between a user system and a plurality of communication networks that provide services to a user, the method comprising:

receiving a user login into the access server;

processing the user login to determine if the user is allowed access to the access system based on a local database system;

providing access to the access system to the user in response to the determination that the user is allowed access based on the local database system;

generating an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system;

in the second database system, receiving and processing the authorization query to determine whether the user is allowed access;

in the second database system, generating and transmitting an authorization response to the local database system;

receiving and processing the authorization response indicating whether the user is allowed to use the access system from the second database system; and

providing access to the access system to the user in response to the authorization response that allows the user to use the access system.

2. The method of claim 1 further comprising generating and transmitting a login query.

3. The method of claim 1 wherein processing the user login to determine if the user is allowed access to the access system based on a local database system is based on a user access profile.

4. The method of claim 1 further comprising determining if the second database system exists.

5. The method of claim 1 further comprising disconnecting a user's network device from the access server in response to the determination that the user is not allowed to use the access system.

6. The method of claim 1 wherein the login reply includes an access card account code.

7. The method of claim 1 wherein the login reply includes foreign network account information and further comprising identifying the second external database based on the foreign network account information.

8. The method of claim 1 further comprising:

receiving a request for access into a service control point;

57

processing the request for access to determine the destination for the request;

generating and transmitting a reply to route the request to the access server;

9. The method of claim 1 further comprising logging contract and settlement information.

10. An access system for providing access between a user system and a plurality of communication networks that provide services to a user, the access system comprising:

an access server connected to the user system and the plurality of communication networks and configured to receive and transmit a user logon from the user system to a local database system;

the local database system connected to the access server and configured to receive the user logon, process the user logon to determine if the user is allowed access to the access system based on the local database system, provide access to the access system to the user in response to the determination that the user is allowed access based on the local database system, generate an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system, receive and process an authorization response indicating whether the user is allowed to use the access system from the second database system, and provide access to the access system to the user in response to the authorization response that allows the user to use the access system; and

the second database system connected to the local database system and configured to receive and process the authorization query to determine whether the user is allowed access and generate and transmit the authorization response for the local database system.

11. The access system of claim 10 wherein the access server is configured to generate and transmit a logon query.

12. The access system of claim 10 wherein the local database system is configured to process the user logon to determine if the user is allowed access to the access system based on a user access profile.

13. The access system of claim 10 wherein the local database system is configured to determine if the second database system exists.

14. The access system of claim 10 wherein the access server is configured to disconnect a user's network device in response to the determination that the user is not allowed to use the access system.

15. The access system of claim 10 wherein the logon reply includes an access card account code.

16. The access system of claim 10 wherein the logon reply includes foreign network account information and the local database system is configured to identify the second external database based on the foreign network account information.

58

17. The access system of claim 10 further comprising:

a service control point connected to the user system configured to receive a request for access, process the request for access to determine the destination for the request, and generate and transmit a reply to route the request to the access server.

18. The access system of claim 10 wherein the local database system is configured to log contract and settlement information.

19. A software product for providing access between a user system and a plurality of communication networks that provide services to a user, the software product comprising:

database software operational when executed by a processor to direct the processor to receive the user logon, process the user logon to determine if the user is allowed access to an access system based on a local database system, provide access to the access system to the user in response to the determination that the user is allowed access based on the local database system, generate an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system, receive and process an authorization response indicating whether the user is allowed to use the access system from the second database system, and provide access to the access system to the user in response to the authorization response that allows the user to use the access system; and

a software storage medium operational to store the database software.

20. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to process the user logon to determine if the user is allowed access to the access system based on a user access profile.

21. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to determine if the second database system exists.

22. The software product of claim 19 wherein the logon reply includes an access card account code.

23. The software product of claim 19 wherein the logon reply includes foreign network account information and the database software is operational when executed by the processor to direct the processor to identify the second external database based on the foreign network account information.

24. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to log contract and settlement information.

* * * * *



US006636259B1

(12) **United States Patent**
Anderson et al.

(10) **Patent No.:** **US 6,636,259 B1**
(45) Date of Patent: **Oct. 21, 2003**

- (54) **AUTOMATICALLY CONFIGURING A WEB-ENABLED DIGITAL CAMERA TO ACCESS THE INTERNET**
- (75) Inventors: **Eric C. Anderson**, San Jose, CA (US);
Robert Paul Morris, Raleigh, NC (US)
- (73) Assignee: **IPAC Acquisition Subsidiary I, LLC**,
Peterborough, NH (US)
- (*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 32 days.
- (21) Appl. No.: **09/625,824**
- (22) Filed: **Jul. 26, 2000**
- (51) Int. Cl.⁷ **H04N 5/232**
- (52) U.S. Cl. **348/211.3; 348/211.12; 348/14.04**
- (58) **Field of Search** **348/14.01, 14.02, 348/14.04, 14.08, 14.09, 552, 143, 156, 157, 211.3, 211.12; 709/322**
- (56) **References Cited**

U.S. PATENT DOCUMENTS

- 5,430,827 A * 7/1995 Rissanen 704/272
5,737,491 A * 4/1998 Allen et al. 704/270
5,905,736 A * 5/1999 Ronen et al. 370/546
6,064,671 A * 5/2000 Killian 370/389
6,067,571 A * 5/2000 Igarashi et al. 709/232

- 6,167,469 A * 12/2000 Safai et al. 710/62
6,226,752 B1 * 5/2001 Gupta et al. 713/201
6,269,481 B1 * 7/2001 Perlman et al. 717/11
6,502,195 B1 * 12/2002 Colvin 713/202

* cited by examiner

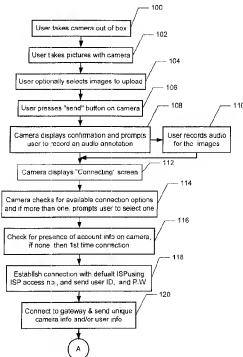
Primary Examiner—Wendy R. Garber
Assistant Examiner—Jacqueline Wilson

(74) *Attorney, Agent, or Firm*—Sawyer Law Group LLP

(57) ABSTRACT

The present invention provides a method and system for automatically establishing a user account and for configuring a hand-held electronic device for accessing a site on a public network using the user account. The method and system include establishing a connection to a website server, which is responsible for establishing and maintaining website accounts. The presence of account information on the electronic device camera is then checked, and if none are found, it is determined that this is a first-time connection and information uniquely identifying the electronic device is sent to the website server so that the server can set-up the account information based on the electronic device information. The server then sends user account information to the device, including an account ID and password. The user account information is then stored on electronic device for use the next time the electronic device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network.

22 Claims, 6 Drawing Sheets



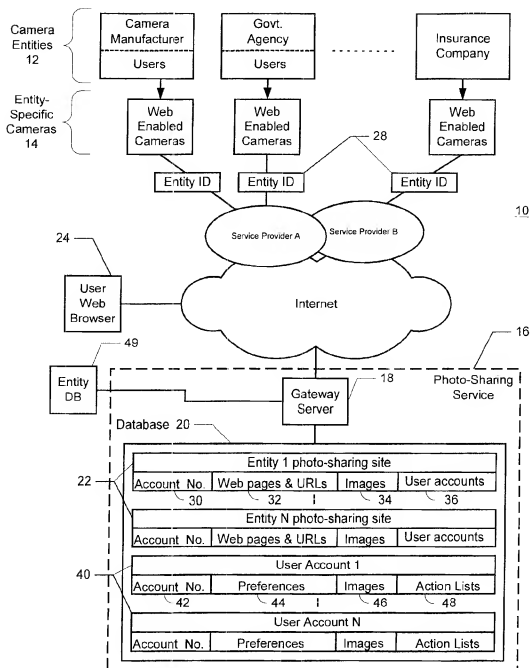


FIG. 1

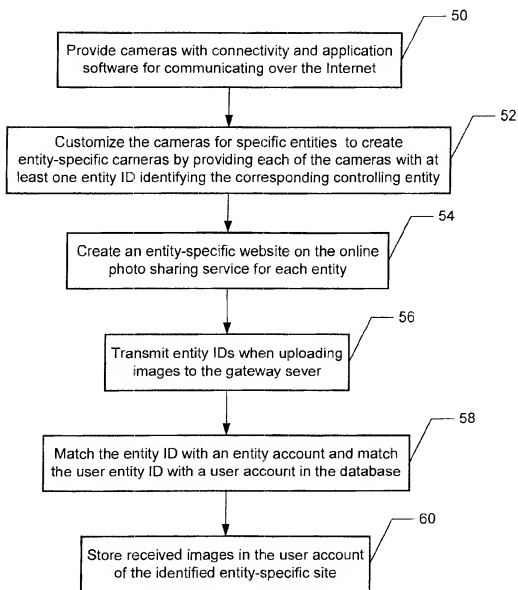


FIG. 2

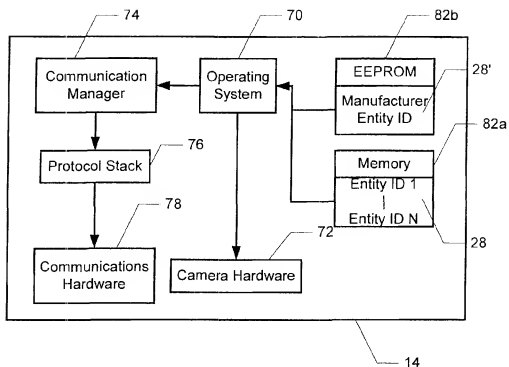


FIG. 3

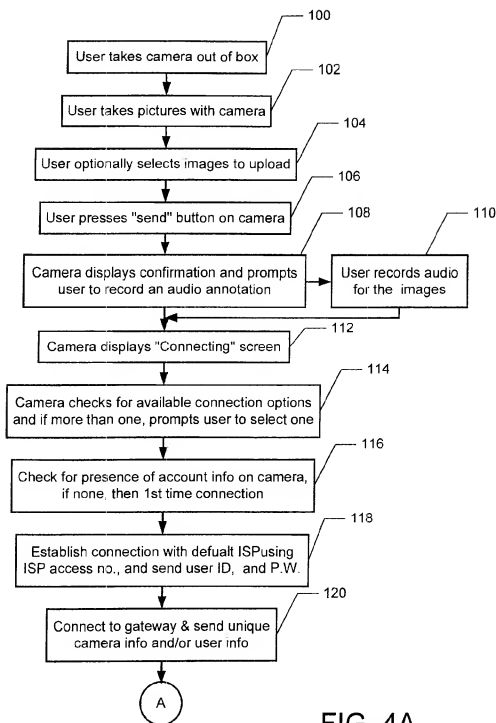


FIG. 4A

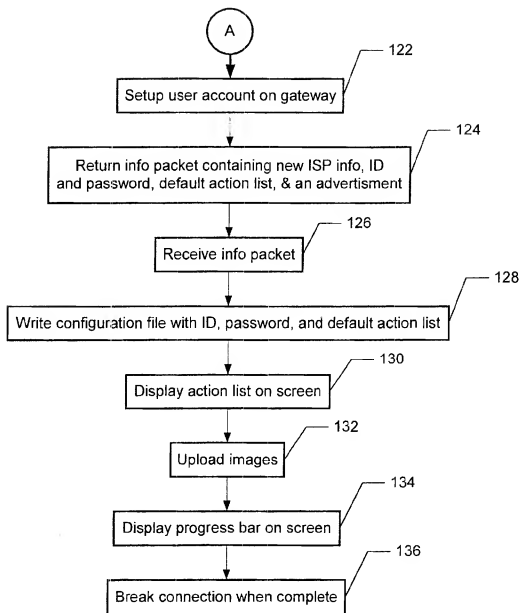


FIG. 4B

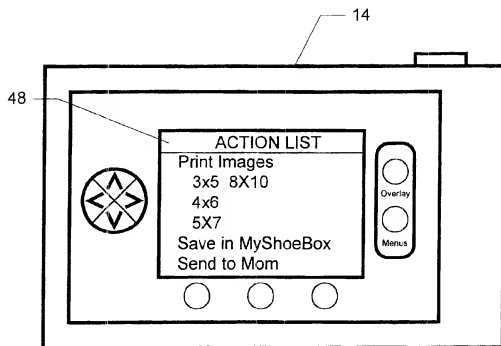


FIG. 5

1

AUTOMATICALLY CONFIGURING A WEB-ENABLED DIGITAL CAMERA TO ACCESS THE INTERNET

CROSS-REFERENCE TO RELATED APPLICATIONS

The present invention is related to co-pending U.S. patent application Ser. No. 09/625,398 entitled "Method and System For Hosting Entity-Specific Photo-Sharing Websites For Entity-Specific Digital Cameras"; and to co-pending U.S. patent application Ser. No. 09/626,418, entitled "Method And System For Selecting Actions To Be Taken By A Server When Uploading Images," which are assigned to the assignee of the present application and filed on the same date as the present application.

FIELD OF THE INVENTION

The present invention relates to a method and system for customizing digital cameras to upload images to an entity-specific photo-sharing websites, and more particularly to automatically configuring a web-enabled digital camera to access the Internet.

BACKGROUND

As the popularity of digital cameras grows, the desire of digital camera users to share their images with others will also continue to grow. New digital camera owners typically try to share their images based on the paradigm of film cameras, in which images are printed on paper and then placed into a photo album. The most straightforward approach to do this with a digital camera is to connect the digital camera directly to a printer to create the prints, and then manually insert the images into a photo album. Users often find this process somewhat complicated and restrictive because standard printers can only print images in limited sizes and requires particular types of paper. And even after the photo album has been assembled, the printed images are not easily shared with many people.

The best approaches to photo-sharing take advantage of the Internet. One such approach is for users to store the digital images on a PC and then send the images to others using email. Several Internet companies now offer an even more convenient approach by providing photo-sharing websites that allow users to store their images for free and to arrange the images into web-based photo albums. Once posted on a photo-sharing website, others may view the images over the Internet.

While convenient for storing digital images, getting the images to the photo-sharing websites can be challenging for users. Most commonly, users must upload their images from the digital camera to a PC using a cable or IrDA, or by inserting the camera's flash card into the PC. From the PC, the user logs onto the Internet and uploads the images to a photo-sharing website. After uploading the images, the user works on the website to arrange the images into web albums and to add any textual information.

Although today's approach for storing images from a digital camera onto a web photo-sharing website and for creating web photo albums works reasonably well, two problems remain that hinder the mainstream adoption of web-based photo-sharing. One problem is that this approach requires the use of a PC, notebook computer, or PDA. While many digital camera users today have PC's, most of those users are early adopters of technology. There are many other consumers who would purchase a digital camera, but are

2

reluctant to do so because they do not yet own a PC or are intimidated by them.

In an effort to address this problem, the assignee of the present application developed an approach to uploading images to the web that doesn't require the use of a PC. In this approach, an email software application is loaded into a digital camera capable of running software that allows the user to e-mail the images directly from the camera. The user simply connects his or her digital camera to a cellphone or modem, runs the e-mail application, and selects the desired images and the email recipients. The selected images are then sent to the recipients as e-mail attachments.

Although emailing photos directly from the camera allows users who do not own a PC to share images over the Internet, these users must still establish accounts with both an Internet service provider (ISP) and the photo-sharing website before being able to post their images. These accounts must also be set-up by PC users as well. For techno savvy users who use a PC to upload the images to the photo-sharing website, establishing the accounts may not be a bother, but even these users may not always have their PC's handy, such as when on vacation, for instance. And for non-PC users, establishing the accounts by entering account information on the digital camera itself may prove to be a cumbersome, if not a daunting, task.

Accordingly, what is needed is an improved method for uploading images from a digital camera to a photo-sharing website on the Internet. In order for online photo-sharing to become more mainstream, an approach that doesn't require a PC or PC expertise and that reduces complexity for the user is required. The present invention addresses such a need.

SUMMARY OF THE INVENTION

The present invention provides a method and system for automatically establishing a user account and for configuring a hand-held electronic device for accessing a site on a public network using the user account. The method and system include establishing a connection to a website server, which is responsible for establishing and maintaining website accounts. The presence of account information on the electronic device camera is then checked, and if none are found, it is determined that this is a first-time connection and information uniquely identifying the electronic device is sent to the website server so that the server can set-up the account information based on the electronic device information. The server then sends user account information to the device, including an account ID and password. The user account information is then stored on electronic device for use the next time the electronic device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network.

According to the method and system disclosed herein, a user does not have to enter account information in order to establish the ISP connection or the website account before accessing the public network with the electronic device.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG 1 is a block diagram illustrating an online photo-sharing system in accordance with a preferred embodiment of the present invention.

FIG 2 is a flow chart illustrating a business method for hosting entity-specific photo-sharing websites for entity-specific image capture devices in accordance with a preferred embodiment of the present invention.

3

FIG. 3 is a diagram showing a preferred embodiment of the connectivity and application software of the camera.

FIGS. 4A and 4B illustrate a flow chart of a process for automatically configuring a web-enabled digital camera to access the Internet in a preferred embodiment of the present invention.

FIG. 5 is a block diagram illustrating an example action list 48 displayed on the LCD screen of the camera.

DETAILED DESCRIPTION

The present invention relates to an automatic system for uploading images from a digital camera to entity-specific photo-sharing websites and for automatically establishing accounts. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

FIG. 1 is a block diagram illustrating an online photo-sharing system 10 in accordance with a preferred embodiment of the present invention. The system includes a plurality of camera controlling entities 12 that produce, own, or otherwise control a set of digital cameras 14, and an online photo-sharing service 16. The online photo-sharing service 16 includes a gateway server 18 and an entity/account database 20. The various camera controlling entities 12 contract with the photo-sharing service 16 to transparently host customized photo-sharing websites 22 for each entity, which are referred to herein as entity-specific photo-sharing websites 22. The entity-specific photo-sharing websites 22 are each accessible to the user through the entity's existing Internet site (not shown), and thus appear to users as though the entity-specific photo-sharing websites 22 are hosted by the corresponding entity. According to a preferred embodiment of the present invention, the cameras 14 for a particular entity are customized for that entity to create entity-specific cameras 14, such that when the cameras 14 connect to the Internet, the cameras 14 automatically upload their images to the photo-sharing website of the corresponding entity. In a further aspect of the present invention, the photo-sharing service 16 automatically stores the images in a web album, which is viewable over the Internet by a user's web browser 24.

As used herein, a camera controlling entity 12 is any entity that makes, owns, sells, or controls digital cameras 14, and therefore includes, camera manufacturers, companies, retailers, and end-users. One or more combination of these entities 12 may either contract with the photo-sharing service 16 to provide entity-specific websites 22 for their cameras 14, or have entity information transmitted to the photo-sharing service 16 from the cameras 14. Therefore, a camera controlling entity 12 may include a single entity 12 or a hierarchical relationship of entities 12.

An example of a single entity 12 includes an insurance company that contracts with the photo-sharing service 16 to have all digital cameras 14 used by their agents to transmit their images to a customized insurance photo-sharing website. Examples of a hierarchical relationships of entities 12 includes a camera manufacturer, such as Nikon, that contracts with the photo-sharing service 16 to have all Nikon digital cameras 14 transmit their images to the customized

4

Nikon photo-sharing website. Since the images of different users must be distinguished, each user of a Nikon camera 14 would also constitute an entity within the Nikon website so that the images from different users can be distinguished. Other examples of hierarchical entity relationships include a retailer and its consumers, a real estate agency and its agents, community groups and its members, and government agencies and its employees, for instance.

In a preferred embodiment, the cameras 14 are customized for their respective entities 12 by providing the cameras 14 with software for transmitting entity ID information 28 identifying its controlling entity 12 to the photo-sharing service 16. The photo-sharing service 16 in conjunction with the gateway server 18 and the entity/account database 20 hosts the entity-specific photo-sharing websites 22. Each entity-specific website 22 is identified in the database 20 by an entity account number 30 and includes the web pages and URIs 32 comprising the website, the images and web albums 34 stored on the website, and the user account numbers 36 of authorized users. The database 20 also includes user accounts 40, each of which comprises a user account number 42, user preferences 44, the user's images 46, and action lists 48, explained further below.

The gateway server 18, which communicates with the cameras 14 during image uploading, receives one or more entity IDs 28 from each camera 14 and matches the entity ID 28 with an entity account 30 in the database 20. The images are then automatically associated with the photo-sharing website 22 of the identified entity 12 and/or the identified user.

After the images are uploaded, a user of the camera 14 may visit the online photo-sharing website 22 over the Internet to view the images via a web browser 24. Since the photo-sharing websites 22 are transparently hosted by the photo-sharing service 16, each photo-sharing website 22 appears as though it is hosted by the entity itself, rather than the third party service.

In one embodiment, the cameras 14 may connect to the Internet via a service provider 26, which may include a wireless carrier and/or an Internet service provider (ISP) that is capable of servicing many devices simultaneously. In a preferred embodiment, each of the cameras 14 is provided with wireless connectivity for connecting to the Internet, and are therefore so called "web-enabled" devices, although a wired connection method may also be used.

The cameras 14 may be provided with wireless connectivity using anyone of a variety of methods. For example, a cellphone may be used to provide the digital camera 14 with wireless capability, where the camera 14 is connected to the cellphone via a cable or some short-range wireless communication, such as Bluetooth. Alternatively, the camera 14 could be provided with built-in cellphone-like wireless communication. In an alternative embodiment, the digital camera 14 is not wireless, but instead uses a modem for Internet connectivity. The modem could be external or internal. If external, the camera 14 could be coupled to modem via any of several communications means (e.g., USB, IEEE1394, infrared link, etc.). An internal modem could be implemented directly within the electronics of camera 14 (e.g., via a modem ASIC), or alternatively, as a software-only modem executing on a processor within camera. As such, it should be appreciated that, at the hardware connectivity level, the Internet connection can take several forms. Hence, it should be appreciated that the present invention is not limited to any particular method of accessing the Internet.

5

In a preferred embodiment, the entity-specific websites 22 are customized to seamlessly integrate into the entity's existing website by following the look and feel of the entity's existing website. The entity-specific websites 22 are hosted on the photo-sharing service 16, but a link to the entity-specific websites 22 may be provided on the homepage of the corresponding entity's existing website. Thus, in order to view a web album on an entity-specific website 22, the user must visit the entity's existing website and click the link to the entity-specific website 22, where the user's browser 24 will be transparently directed to the photo-sharing service 16 and be provided with the web pages 32 of the entity-specific website 22.

As an example of the operation of the photo sharing system 10, consider the following scenario. Assume that Minolta and Nikon are entities 12 that have contracted with the photo-sharing service 16, and that the photo-sharing service 16 hosts a photo-sharing website 22 for Minolta and a photo-sharing website 22 Nikon. The Minolta cameras 14 would be provided the entity ID 28 for Minolta and the Nikon cameras 14 would be provided the entity ID 28 for Nikon. When the Minolta and the Nikon cameras 14 send sets of images to the photo-sharing service 16, the gateway server 18 would distinguish the cameras 14 by the entity IDs 28 and would direct the set of images received from Minolta cameras 14 to Minolta's photo-sharing website, and would direct the images from Nikon cameras 14 to Nikon's photo-sharing website. To view the images, the owners of the cameras 14 would use a browser 24 on their PC or PDA to visit the URL of the Minolta or Nikon photo-sharing websites 22. In one preferred embodiment, the photo-sharing service 16 sends the URL of the entity-specific website 22 directly to the camera 14 for display to inform the user of the address.

According to the present invention, the photo-sharing service 16 provides business-to-business and business-to-consumer business models. The service is business-to-business because the service provides companies, such as camera manufacturers, with a complete end-to-end solution for their cameras 14. The solution includes customized software for their cameras 14 for sending images over the internet, and an internet website for storing images from those cameras 14 on a branded website that appears to be hosted by the company. The service is business-to-consumer because it allows users of digital cameras 14 with an automatic solution for uploading captured images from a digital camera 14 to an online photo-sharing website, without a PC.

According to one preferred embodiment of the present invention, the photo-sharing service 16 provides a method of doing business whereby the photo-sharing service 16 shares revenue based on the hierarchical relationship of the entities 12. For example, if the photo sharing service 16 charges a fee for receiving and/or storing the images received from the entity-specific cameras 14, then the photo sharing service 16 may share a portion of the fee with the manufacturer and/or third party supplier of the camera 14 that uploaded the images, for instance. Revenue may also be shared with the wireless service provider providing the connection with the photo sharing service 16.

FIG. 2 is a flow chart illustrating a business method for hosting entity-specific photo-sharing websites for entity-specific image capture devices, such as digital cameras, in accordance with a preferred embodiment of the present invention. First, the cameras 14 are provided with connectivity and application software for communicating over the Internet in step 50. In a preferred embodiment, this step is

6

performed during camera 14 manufacturing to provide off-the-shelf web enabled cameras 14. The cameras 14 are also customized for specific entities 12 to create entity-specific cameras 14 by providing each of the cameras 14 with at least one entity ID 28 identifying the corresponding controlling entity in step 52.

Referring now to FIG. 3, a diagram showing the preferred embodiment of the connectivity and application software of the camera 14 and the entity ID 28 information is shown. Preferably, the camera 14 includes a microprocessor-based architecture that runs an operating system 70 for controlling camera hardware 72 and overall functionality of the camera 14 (e.g., taking pictures, storing pictures, and the like). An example of such an operating system 70 is the Digita™ Operating Environment developed by the assignee of the present application. The camera 14 also includes communication manager 74 software, and a TCP/IP protocol stack 76, that enables communication via the internet, as is well-known in the art. The entity ID information 28 and captured images may be stored in one or more types of memories 82.

For hierarchical entity relationships, the cameras 14 are provided with hierarchical entity IDs 28; one entity ID 28 identifying the entity, and a second entity ID 28 identifying the end-user. Whether there are one or more entity IDs 28, the entity ID 28 of the camera manufacturer may always be provided. Camera 14 customization may occur either during manufacture or anytime thereafter. In a preferred embodiment, the manufacturer entity ID 28 is provided at the time of manufacturing and is stored in an EEPROM 82b, while the entity IDs 28 for other entities 12, such as companies and end-users, are loaded into the camera 14 subsequent to manufacturing and are stored in flash memory 82a or the EEPROM 82b.

Customization that occurs subsequent to manufacture may be implemented using several methods. The first method is to manufacture the cameras 14 with an application programming interface (not shown) for accepting a subsequently loaded software application that specifies the entity ID's 28. The application may come preloaded on a flash card, which is then inserted into the camera 14 by the user and stored in flash memory 82a. The application may also be wirelessly beamed into the camera. When executed in the camera, the software application transmits the appropriate entity IDs 28 to the gateway server 18.

The second method is to load a small file in the camera 14 specifying the entity IDs 28 from a removable memory or from a PC, and storing the file in a system folder within the camera's flash memory 82a. The camera 14 then accesses the file when an Internet connection is established. In a preferred embodiment, the communication manager 74 automatically extracts the manufacturing ID 28 and the entity ID 28 and transmits them to the gateway server 18. In this embodiment, the entity ID 28 is also stored in the EEPROM 82b and is factory set to zero (empty). Thus, unless the entity ID 28 is set, the manufacturing ID 28 may default as the highest controlling entity.

If, for example, a third party developer X contracts to provide custom camera software for camera manufacturer Z, then a custom entity ID will be issued for developer X and developer X will place the custom entity ID into the EEPROM 82b. Developer X is now a controlling entity 12, and may specify to the photo-sharing service 16 that a developer X entity-specific photo-sharing site 22 or developer X's own website be the destination for the uploaded images.

7

The protocol stack 76, under direction of the communications manager interfaces with the communications hardware 78 of camera. The protocol stack 76 includes software APIs and protocol libraries that interface with the communication manager 74, and communication hardware interface drivers that interfaces directly with the various communication hardware 72 the camera 14 must function with (e.g., a Bluetooth transceiver, etc.). The communication manager 74 communicates with operating system 70 and the IP protocol stack 76 to establish an Internet connection ant to transmit the entity ID 28 information and images from the memories 82a and 82b to the photo-sharing service 16.

In an alternative embodiment, rather than loading entity ID's 28 into the camera, a combination of the camera's serial number and the make and model number of the camera may be used as the entity ID 28. Entity specific cameras 14 may then be distinguished by providing a mapping of the camera serial numbers and product IDs to specific entities 12 in the database 20.

Although the camera 14 has been described in terms of a software-based customization solution, those with ordinary skill in the art will readily recognize that the camera 14 may also be provided with a hardware-based solution.

Referring again to FIG. 2, before or after camera customization, the entity-specific websites 22 are created for each entity contracting with the photo-sharing service 16 in step 54. Customization requires storage of entity information in the entity/account database 20 and creating and storing web page elements comprising the entity-specific photo-sharing website in the database 20. The entity-specific information stored in the database 20 may also include service levels, and enabled features for the entity-specific website 22. Features are components or services that may be provided on websites by the photo-sharing service 16, such as search functions, and online printing, for instance, but may be selectable by each entity for its own website. As an example, company X may provide customized cameras 14 for its employees, but may not wish to allow employees to print images from the company X photo website for security reasons. If so desired, company X may have the photo service disable this feature from their particular website.

In a preferred embodiment, the entity-specific websites 22 are not created from scratch, but are created by modifying a preexisting template. The template may include several different sections, such as A, B, C and D, for instance. Assuming for example that the template used to create a website for Nikon, and section A is used to specify the name of the entity then the name Nikon would be inserted into that section. Other entity-specific content would be used to fill out the remaining sections. The Web pages comprising the Nikon specific photo-sharing website would then be provided with URL's unique to that website. The entity's regular website would be modified to include a link to the entity's photo-sharing website 22. In addition, the entity-specific photo-sharing website would include a link back to the entity's website. Entities 12 may have entity photo-sharing websites 22 created for them in one of two ways; automatically by logging into the photo-sharing service 16 and manually customizing the templates, or by having the entity photo-sharing website created for them.

Referring still to FIG. 2, when a camera 14 establishes an internet connection with the gateway server 18, the camera 14 transmits its entity IDs 28 and/or user entity ID 28 when uploading user selected images to the gateway server 18 in step 56. In response, the gateway server 18 matches the entity ID 28 with an entity account in the database 20 and

8

matches the user entity ID 28 with a user account 40 in the database 20 in step 58. The images received are then stored in the user account 40 of the identified entity-specific website 22 in step 60.

Referring again to FIG. 1, each user account 40 in the database 20 may also include one or more action lists 48. According to the present invention, an action list 48 includes one or more items representing actions that the gateway server 18 should take with respect to uploaded images, such as where to store and/or send the images from a particular user or camera, for instance. As explained further below, the action list 48 stored on the database 20 under a user's account 40 are automatically downloaded to the user's camera 14 during a connection with the gateway server 18 and stored on the camera 14. When the user initiates an image upload, the action list 48 is displayed to the user so the user may easily select what actions the gateway server 18 should take with respect to the images by selecting the displayed action list items.

Examples of action list items include specifying that the uploaded images should be stored on the entity-specific photo website, sending the images to a list of email addresses, or even performing some type of analysis or calculation on the image data, for instance.

In a further aspect of the present invention, an action list item is not limited to, instructing the gateway server 18 to perform actions only within the photo-sharing service 16. Rather, an item in the action list 48 may also instruct the gateway server 18 to perform actions outside of the photo-sharing service 16, such as storing the images in an external database 49 of the entity 12. For instance, in the example where the entity 12 is a company, some users of the company's cameras 14 could have action lists 48 instructing the gateway server 18 to store uploaded images to the company's database, rather than to the company's photo-sharing site 22. Based on the action lists 48 and customization, the gateway server 18 may be programmed to automatically perform predefined tasks, such as creating new web albums, or a new page within an existing album, parse the images to extract sound files or other metadata, print images and mail them to designated addresses, and so on.

In a preferred embodiment, the action lists 48 may be created via several methods. In one method, the action list is created by the photo-sharing service 16 the first time the user's camera establishes a connection. That is, a default action list 48 is automatically created based on the entity ID when a user account 40 is first created. In a hierarchical entity relationship where the entity 12 is a company, a default action list 48 may be created to implement a workflow specified by the entity 12. In a hierarchical relationship where the entity 12 is camera manufacturer, for instance, a default action list 48 may be created instructing the gateway server 18 to store the user's images in a simulated "shoebox" on the entity-specific photo-sharing site 20. The user may then go online and create albums from the images in the shoebox as desired.

Another method is for the user to create the action list 48 online on the entity-specific photo-sharing site 20. The action list 48 may be created manually on the website 20 by the user navigating to the site 20 using a web browser 24, accessing her account, and manually creating the action list 48 or editing the action list 48 on the entity-specific site 22. The action list 48 may also be created automatically on the website 20 in response to user actions performed on the website, such as printing images, or creating a web album.

9

Alternatively, after performing an action, the user may be prompted whether they would like this action added to his or her action list 48. If so, the user clicks a check-box and the item is added the action list 48. In a preferred embodiment, any action list 48 created and edited on the photo-sharing site 20 are downloaded to the camera every time the camera 14 connects to the photo-sharing service 16 and made available for user selection on the camera 14 during the next upload.

Yet another method for creating an action list 48 is to allow the user to create the action list on the camera 14. The user may manually create an action list 48 by "typing" in predefined items on the camera. The user may also type in an email address as an action list item whereby when that item is selected, the uploaded images are stored as a web album on the entity-specific photo-sharing website 22 and the server 18 sends a notification to the specified recipient containing the URL to the web album page.

A method and system for hosting web-based photo-sharing websites and for customizing digital cameras to upload images to the entity-specific photo-sharing websites has been disclosed. According to the present invention, users of the customized cameras 14 can upload images to the Internet for storage and web photo album creation without the use of a PC.

In one embodiment described above, the present invention assumes that an ISP account has been established with the digital camera's service provider, and that users of cameras 14 belonging to a certain entity 12 may use the cameras 14 to upload images to the website of the entity 12. However, two problems with account setup remain. One problem is that just as with a PC and PDA, the user must first establish an ISP account before the camera 14 can establish Internet communication. The second account problem is that most websites, including the photo sharing sites 22, may require each user to establish a unique account before using the site to distinguish one user from another. Before being able to connect the web-enabled camera 14 to the Internet, the user must establish these two accounts by either entering account setup information on a PC or entering account setup information on the camera 14. Neither alternative is a convenient alternative for people who do not have the time nor inclination to do so.

In a further aspect of the present invention, the cameras and the photo sharing site are provided with software for automatically creating Internet and photo-sharing website accounts for each camera 14 upon first use, without requiring the user to first enter account information on a PC or on the camera 14.

Referring now to FIGS. 4A and 4B, a flow chart illustrating a process for automatically configuring a web-enabled digital camera to access the Internet in a preferred embodiment of the present invention. Although the process will be described in terms of automatically establishing Internet accounts for a digital camera without requiring the user to enter information, those with ordinary skill in the art will readily recognize that the present invention may be used to automatically establish Internet accounts for any type of portable electronic device.

The process assumes that a user has just acquired a digital camera 14 customized as described above, and has just taken the camera 14 out of its box in step 100. After taking pictures with the camera in step 102, the user may review the images in the camera's LCD screen and optionally select a set of images to upload to the photo sharing service 16 in step 104. The user then presses a "send" button on the camera in step 106 to upload the images.

10

In response, the camera displays a confirmation dialog screen on the camera and prompts the user to record an audio annotation for the images or to continue in step 108. The user may then choose to record audio for the images in step 110. After choosing to continue, or after recording audio, the camera displays a "connecting" dialog screen in step 112 to indicate to the user that the camera is establishing an Internet connection. At the same time, the camera checks for available connection options in step 114, and if more than one is found, the camera prompts the user to select one of the connection options. For example, the camera may be within range of a Bluetooth-equipped printer and a cellphone, so the user will be prompted to choose which device the camera should establish communication with.

The camera then checks for the presence of account information on the camera in step 116, and if there are none, the camera assumes that this is a first-time connection. According to the present invention, in order to allow the camera to make a first-time Internet connection, the camera is provided with default Internet service provider (ISP) information during manufacturing, including an ISP access number, and user ID and password (if required). The camera establishes connection with the default ISP in step 118 by dialing the preloaded access number, and by sending the preloaded user ID and password to the ISP. This special account may be configured so that the camera can only connect to the gateway server 18 (no other IP addresses may be allowed).

After connecting with the ISP, the camera connects to the gateway server 18 and sends unique camera information and/or user information in step 120. In a preferred embodiment of the present invention, a combination of the camera's serial number and the make and model number of the camera may be sent as the unique camera information. In another preferred embodiment, the user's e-mail address may be sent as the unique camera or user information.

Continuing with FIG. 4B, the gateway server 18 uses the unique camera information to set up a user account 40 in step 122. After creating the user account 40, the gateway server 18 returns an information packet to the camera containing new ISP information (if needed), an account ID, and an account password in step 124. The information packet may also contain a default action list specifying what actions should be taken with respect to the images, an advertisement for display on the camera, and the URL of the entity-specific website 22.

It should be noted that if the camera is used in conjunction with an IP direct phone or is provided with a phone number for connected to a dedicated server where the user is not billed separately for the ISP connection, then the steps of providing the camera with default ISP info and returning new ISP info, may be omitted.

The camera receives the information packet in step 126, and writes a configuration file to memory 82a containing the ID, password, and default action list in step 128. The camera then displays the action list on the camera's LCD screen for selection by the user in step 130. The camera may optionally display the user's account information as well.

In an alternative preferred embodiment where security is a concern, the camera 14 first logs off the special ISP and the gateway accounts, and then reconnects using new ISP and gateway accounts in order to retrieve the action lists 48.

FIG. 5 is a block diagram illustrating an example action list 48 displayed on the LCD screen of the camera 14. The action list 48 is shown displaying three major options; printing the uploaded images, saving the uploaded images in

11

the user's shoebox, and sending the images to Mom. Under the printing option, the user may select from various size prints. Rather than nested menu categories as shown under the printing option, the action list 48 may be displayed with each action listed as a separate item (e.g., "Send 4x8 prints to Mom", "Send 5x7 prints to me").

Referring again to FIG. 4B, after the user selects one or more actions from the action list 48, the camera begins to upload the images along with the selected actions in step 132 and displays a progress bar on the screen in step 134. In one preferred embodiment, the camera may also display the advertisement sent in the information packet from the gateway server 18. The advertisement may advertise the controlling entity 12, the entity's photo-sharing site 22, or the photo sharing service 16. After all the images are uploaded and associated with the user's account 40, the camera breaks the connection with the gateway server in step 136. At this point, the camera 14 may also display the URL of the entity-specific website 22 to the user.

The next time the user uploads images, the camera will use any new ISP information received to connect to the Internet and will use the account ID and password written to memory 82a when connecting to the gateway server 18. Thus, by using unique camera information, such as the serial number, to establish a web site account upon first use, the present invention eliminates the need for the user to type in information to establish a web site accounts.

To further explain the present invention from a user interaction point of view, consider the following scenario where a user named Jack has just purchased a digital camera 14 from a store and unpacks the camera from the box. There is a "Quick Start Guide" which guides him in getting started. Jack pops in the batteries, sets the date and time, and takes some pictures of his dog and his new baby. Jack can see the pictures have come out well on the small LCD, and now wants to try sharing them with his parents.

The Quick Start Guide says to select the photos to be sent using the "Select" button, and then press the "Send" button. So, Jack navigates to each of the baby pictures, selects them, and then presses the "Send" button. Instantly, a dialog comes up on the LCD screen: "No Receiving Device Found! Please turn on your phone or other connecting device." Oops! Jack pulls out his cell phone, and turns it on. He presses the "Continue" button. Jack does not see this happen, but the camera now "discovers" the cell phone, and immediately presents another dialog: "4 Images Selected. Press Record to add a Sound Note, or Continue to send". Although Jack finds the proposition of recording sound intriguing, he decides to skip it and presses the "Continue" button. Immediately, the dialog is replaced with a "Connecting . . ." dialog.

Shortly, another dialog appears: "Your camera serial number is 38147. Please write this down. You will need it to access your web photo albums". Jack writes the number down on the spot provided in the Quick Start Guide, and presses OK.

Another "Connecting . . ." dialog appears, and is then quickly replaced by another dialog, which says "A free, temporary account has been set up for you at www.photosharing-service.com/new_accounts. You will need your camera serial number to access your photos and complete the setup of your account. Please complete the account setup within 30 days". Jack writes down the URL in the space provided in the Quick Start Guide, and presses the OK button. Jack doesn't know it, but during this dialog, the camera has begun transmitting his images and is already partially complete.

12

A new dialog comes up, with a progress bar. Jack is surprised to see that the transmission is already almost ½ done. Below the progress bar, the dialog says "Press 'Continue' to use camera during photo transmission, or wait for progress bar to complete". Jack is interested in watching how fast his images are transmitted, and decides to watch the progress bar complete. A "Transmission Successful" dialog appears. Jack presses the OK button. The camera returns to the review mode.

Jack is pretty excited—he just sent four baby pictures to the Internet. Jack then decides to see what happened to his images so he turns on his PC. After connecting to the Internet, Jack types in the URL from the Quick Start Guide. A Photo-sharing service web page appears, welcoming Jack to the photo-sharing service. After looking briefly at the welcome page, Jack types in his serial number from the Quick Start Guide, and selects his camera's model number from a pop-up menu. Jack clicks on a "Submit" button on the web page.

Jack now sees a page which shows thumbnails of the baby pictures he just sent, the page is entitled "My Shoebox". The page explains to Jack that he is looking at his on-line digital photo shoebox. Since the server 18 knows this is Jack's first visit, special help messages may appear. Various options are provided via buttons and text links. One that catches Jack's eye is CREATE WEB PHOTO ALBUM. Jack clicks this button, and works his way through the process of setting up an on-line photo album. This includes selecting photos from the shoebox, as well as selecting layout and style. One of the check-box items Jack is offered is "Make this album a camera Action List Item". Jack doesn't know what that is, so he clicks the Action List link, which brings up a brief description "If you check this box, you will be able to send pictures directly from the camera to this photo album!" Jack finds this interesting, so he closes the description window, and checks the box. Jack also enters the email address for his parents and his wife's parents, so they can be notified to come and see his photo album, which he has entitled "Our First Baby".

One of the buttons Jack does not click is the "Complete Account Setup" button. He knows that he has 30 days to do that chore, and figures he will get back to it later.

Jack's wife arrives home from shopping at this point, and Jack wants to show her how the new camera works. He starts by showing her the baby album on the PC, then decides to take some pictures of them holding the baby. Jack then selects the images, and presses the "Send" button again.

Since his cell phone is still on, sitting on the table a few feet away, the connection goes smoothly and quickly. A new dialog pops up, surprising Jack. It says "Select the destination for your pictures" and offers two choices: "My Shoebox" and "Our First Baby". Jack is amazed, he doesn't realize that the server 18 downloaded his action list to his camera during the connection. Jack decides to select the "Our First Baby" web album as the destination for the images, and clicks OK. The pictures are sent as before. After the transmission has completed, Jack goes to the PC to check on the photo album. When Jack refreshes the album page, he now sees the additional pictures he just sent, along with the pictures he sent before.

Jack spots a "Send Prints" link on the web page, and clicks it. He is led through a selection of print types, mailing addresses, and credit card info to make it possible to send prints. He is offered the option of completing the setup of his account. Jack decides to do that now, and proceeds to fill out the requested information, including his credit card number.

13

Once the account setup is complete, Jack continues the print order. One of the radio button items is "Make prints a separate Action List Item" or "Make prints part of your Action List". Jack remembers something about Action Lists from before, but is not sure what this means. The description says "Making a separate action list for prints allows you to decide in the camera to send to the photo album and send prints or to just send to the photo album." Jack thinks this is cool, and clicks the "Make prints a separate Action List" item. The next time Jack sends photos from the camera, his action list will be updated to present three choices: My Shoebox, Our First Baby, and Our First Baby w/Prints.

The underlying technology supporting this scenario are summarized below. Other functions and features are assumed, but not required for this scenario:

1. Two-way connection between camera and portal
2. Camera metadata included in the request
3. If not using an IP direct connection,
 - 3.1. Default ISP connection info built into the camera for the first connection (country specific)
 - 3.2. Downloaded assigned ISP information from the portal to the camera
4. Software capable of recognizing automatically a set of supported phones and adjusting the protocol to match
5. Action Lists maintained on the server that are automatically downloaded to the camera to update the camera selection list each time a connection is made
6. An On-line shoebox
7. Ability to download files to the camera from the server. The files could be text, GIF, animated GIF, JPG, or even a script or applet. This feature enables the ability to display advertisements on the camera, remind the user of remaining time to complete account setup, make special offers, and indicate limits reached.

A method and system for automatically configuring a web-enabled digital camera to access the Internet has been disclosed. Although the present invention has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations to the embodiments and those variations would be within the spirit and scope of the present invention. For example, although the photo-sharing service has been described as including the gateway server and the database, the database may be located elsewhere. Also, the gateway server may be used to control account information, while one or more other servers may be used to provide the web pages of the entity-specific websites. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.

What is claimed is:

1. A method for automatically establishing a user account and for configuring a hand-held image capture device which includes a digital camera for accessing a site on a public network using the user account, the method comprising the steps of:

- (a) establishing a connection to a website server, the website server being responsible for establishing and maintaining website accounts;
- (b) checking for the presence of account information on the image Capture Device and if none are found, determining that this is a first-time connection and automatically sending image capture device information uniquely identifying the electronic device to the website server so that the server can set-up the account information based on the image capture device information;

14

(c) receiving user account information from the server, wherein the user account information includes an account ID and password; and

(d) storing the user account information on image capture device for use a next time the image capture device accesses the website, whereby the user does not have to enter account information in order to establish the connection with the public network or to establish the website account.

2. The method of claim 1 further including the step of providing a digital camera as the hand-held image capture device.

3. The method of claim 2 further including the step of sending at least a partial serial number as the image capture device information uniquely identifying the electronic device.

4. The method of claim 3 further including the step of uploading captured images to the website server.

5. The method of claim 4 wherein step (a) further including the steps of:

- (i) providing the image capture device with default ISP information; and
- (ii) establishing a connection to an ISP using the default ISP information.

6. The method of claim 5 further including the step of terminating the connection with the ISP when the uploading of the images is complete.

7. The method of claim 6 further including the step of providing the digital camera with a default ISP access number, user ID, and password.

8. A system for automatically establishing a user account and for configuring a hand-held image capture device which includes a digital camera image capture device for accessing a site on a public network using the user account, comprising:

means for establishing a connection to a website server, the website server being responsible for establishing and maintaining website accounts;

means for checking for the presence of account information on the image capture device, and if none are found, determining that this is a first-time connection and automatically sending image capture device information uniquely identifying the image capture device to the website server so that the server can set-up the account information based on the image capture device information; and

means for receiving user account information from the server, wherein the user account information includes an account ID; and

means for storing the user account information on image capture device for use a next time the image capture device accesses the website, whereby the user does not have to enter account information in order to establish the ISP connection or the website account.

9. The system of claim 8 wherein the hand-held image capture device comprises a digital camera.

10. The system of claim 9 wherein the image capture device information uniquely identifying the image capture device includes a serial number of the image capture device.

11. The system of claim 10 wherein the camera uploads captured images to the website server.

12. The system of claim 9 wherein the camera further includes means for providing the image capture device with default ISP information, and means for establishing a connection to an ISP using the default ISP information.

13. The system of claim 12 wherein the digital camera terminates the connection with the ISP upon completion of image uploading.

15

14. The system of claim 13 wherein the camera is provided with a default ISP access number, user ID, and password.

15. A method for automatically establishing a user account and for configuring a digital camera to access a website on a public network using the user account, the digital camera including captured images and communication means for accessing the public network, the method comprising the steps of:

- (e) storing default ISP information on the digital camera, the default ISP information including an access number;
- (f) in response to a user request to upload images to the website, determining whether this is a first-time connection;
- (g) if it is first-time connection, establishing communication with the default ISP by dialing the access number;
- (h) establishing communication with the website and automatically sending a digital camera ID to the website;
- (i) using the digital camera ID to set up a user account on the website;
- (j) sending user account information to the digital camera, the user account including an account ID and an account password;
- (k) storing the user account information on the camera; and
- (l) uploading images to the website and associating the images with the user account, whereby the user does not have to enter information to establish a connection with the public network or to establish the website account.

16. The method of claim 15 further including the step of sending at least a partial serial number as the digital camera ID.

17. The method of claim 16 further including the step of receiving new ISP information from the website.

18. The method of claim 17 further including the step of using the new ISP information a next time the digital camera establishes a connection with the public network.

16

19. A system for automatically establishing a user account and for configuring a digital camera for accessing a website on a public network using the user account, the digital camera including images captured by user and communication means for wirelessly accessing the public network, the system comprising the steps of:

- means for storing default ISP information on the digital camera, the default ISP information including an access number;
- means for determining whether this is a first-time connection in response to a user request to upload images to the website;
- means for establishing communication with the default ISP by dialing the access number;
- means for establishing communication with the website and sending a digital camera ID to the website;
- means for using the digital camera ID to set up a user account on the website;
- means for automatically sending user account information to the digital camera, the user account including an account ID and an account password;
- means for storing the user account information on the camera; and
- means for uploading images to the website and associating the images with the user account, whereby the user does not have to enter information to establish a connection with the public network or to establish the website account.

20. The system of claim 19 wherein the digital camera ID includes a serial number.

21. The system of claim 20 wherein the digital camera receives new ISP information from the website.

22. The system of claim 21 wherein the digital camera uses the new ISP information a next time the digital camera establishes a connection with the public network.

* * * * *



US00635384B1

(12) **United States Patent**
Morris

(10) **Patent No.:** **US 6,353,848 B1**
(45) **Date of Patent:** **Mar. 5, 2002**

(54) **METHOD AND SYSTEM ALLOWING A CLIENT COMPUTER TO ACCESS A PORTABLE DIGITAL IMAGE CAPTURE UNIT OVER A NETWORK**

(75) Inventor: **Robert P. Morris**, Raleigh, NC (US)

(73) Assignee: **FlashPoint Technology, Inc.**,
Peterborough, NH (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/127,514**

(22) Filed: **Jul. 31, 1998**

(51) Int. Cl.⁷ **G06F 15/16**

(52) U.S. Cl. **709/203; 709/200; 709/202; 709/217; 709/219; 709/238; 709/232; 709/245**

(58) **Field of Search** **709/200-203, 709/206, 217-219, 227-229, 231-232, 236-237, 245; 707/10, 200-204; 713/201-202**

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,911,044 A	~	6/1999	Lo et al.	709/203
6,018,774 A	~	1/2000	Mayle et al.	709/203
6,058,428 A	~	5/2000	Wang et al.	709/232
6,085,249 A	~	7/2000	Wang et al.	709/229
6,101,536 A	~	8/2000	Kolani et al.	709/217
6,141,759 A	~	10/2000	Braddy	709/203

FOREIGN PATENT DOCUMENTS

DE 198 08 616 9/1998 H04L/12/16

OTHER PUBLICATIONS

De Albuquerque et al., "Remote Monitoring Over The Internet", Nuclear Instruments & Methods In Physics Research, Section-A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 412, No. 1, Jul. 21, 1998, pp. 140-145.

* cited by examiner

Primary Examiner—Zarni Maung

Assistant Examiner—Bharat Barot

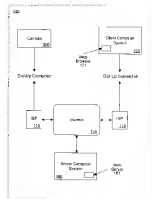
(74) Attorney, Agent, or Firm—Sawyer Law Group LLP

(57)

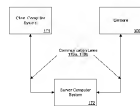
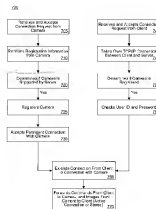
ABSTRACT

A method for accessing a digital image capture unit via a communication network comprising a server computer system and a client computer system communicatively coupled with communication equipment. In one embodiment, the address of the digital image capture unit is registered in an executable program on the server computer system. The executable program is accessed by the client computer system. The executable program connects the digital image capture unit and the server computer system. The executable program communicates commands between the client computer system and the digital image capture unit, such that data captured by the digital image capture unit is transferred to the client computer system via the server computer system.

27 Claims, 11 Drawing Sheets



120



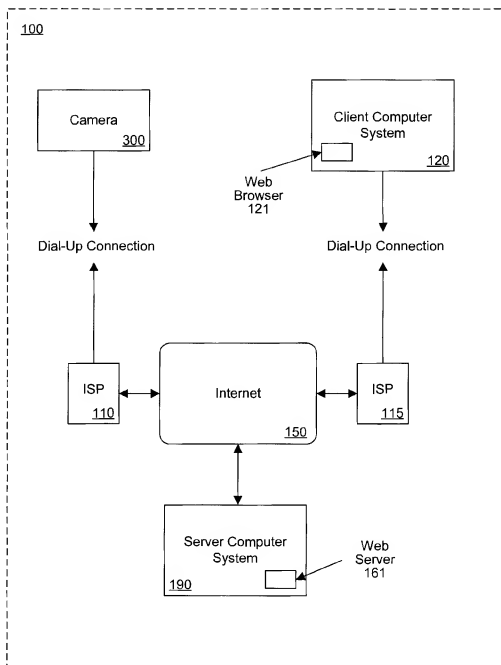


FIG. 1A

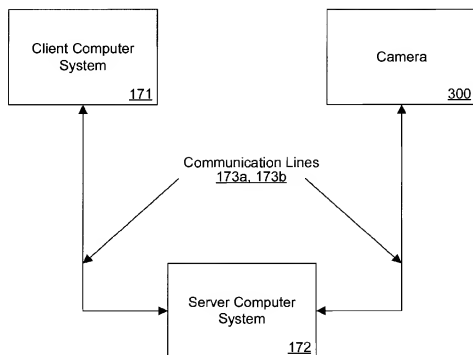
170

FIG. 1B

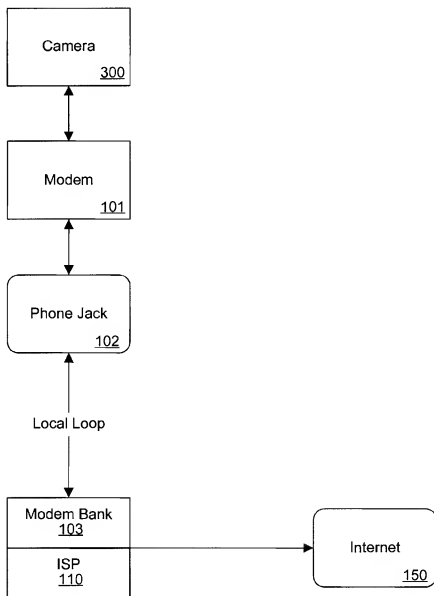


FIG. 1C

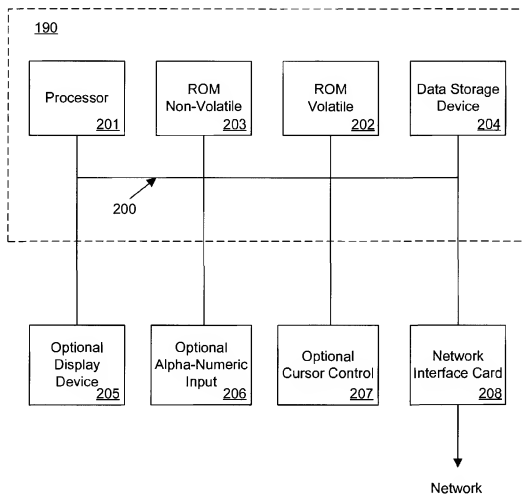


FIG. 2

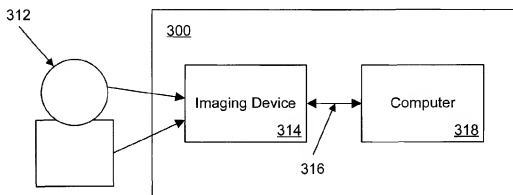


FIG. 3

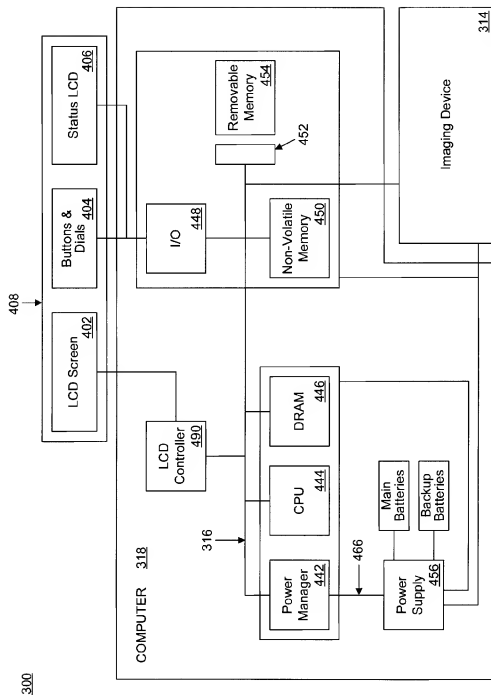


FIG. 4

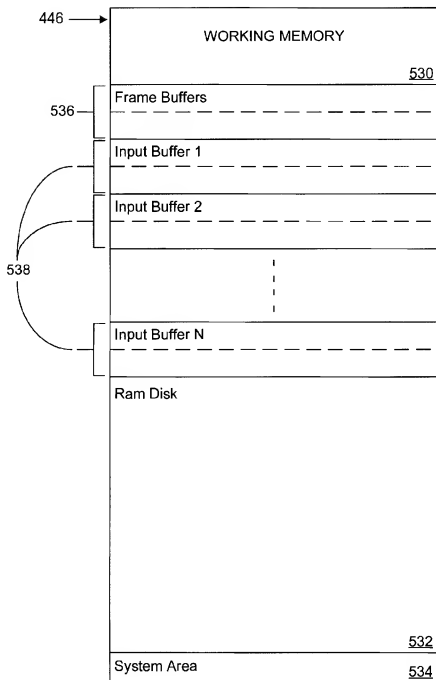


FIG. 5

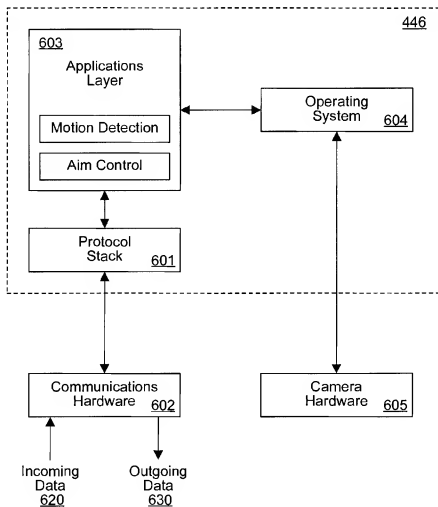


FIG. 6

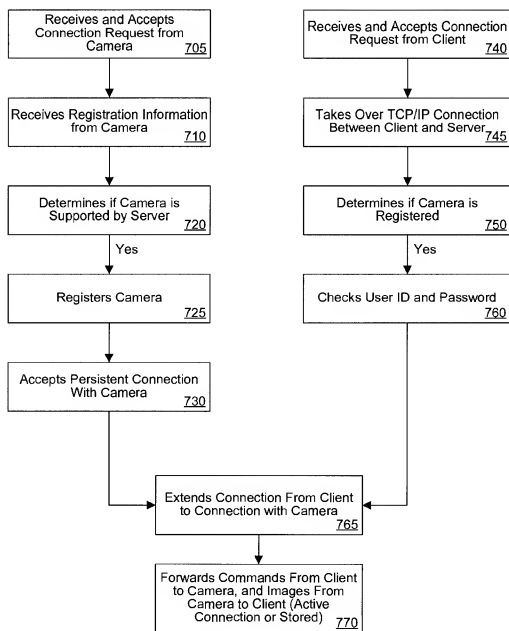
700

FIG. 7

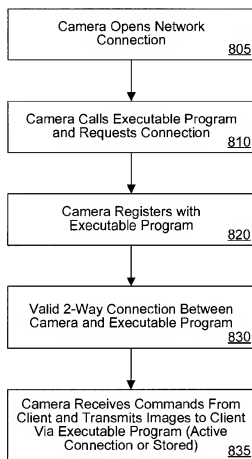
800

FIG. 8

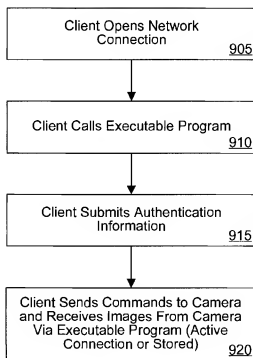
900

FIG. 9

METHOD AND SYSTEM ALLOWING A CLIENT COMPUTER TO ACCESS A PORTABLE DIGITAL IMAGE CAPTURE UNIT OVER A NETWORK

FIELD OF THE INVENTION

The field of the present invention pertains to digital image capture devices. More particularly, the present invention relates to a method for remotely accessing a digital camera via a communication network.

BACKGROUND OF THE INVENTION

Modern digital cameras typically include an imaging device which is controlled by a computer system running a software program. When an image is captured, the imaging device is exposed to light and generates raw image data representing the image. The raw image data are typically stored in an image buffer, where they are processed and compressed by the computer system's processor. Many types of compression schemes can be used to compress the image data, such as the joint photographic expert group (JPEG) standard. After the processor processes and compresses the raw image data into image files, the processor stores the image files in an internal memory or on an external memory card.

Some digital cameras are also equipped with a liquid-crystal display (LCD) or another type of display screen on the back of the camera. Through the use of the LCD, the processor can cause the digital camera to operate in one of two modes, play and record, although some cameras only have a record mode. In the play mode, the LCD is used as a playback screen allowing the user to review previously captured images either individually or in arrays of four, nine, or sixteen images. In the record mode, the LCD is used as a viewfinder through which the user may view an object or scene before taking a picture.

Besides the LCD, user interfaces for digital cameras also include a number of buttons or switches for setting the camera into one of the two modes and for navigating between images in play mode. For example, most digital cameras include two buttons, labeled "-", and "+", that enable a user to navigate or scroll through captured images. For example, if the user is reviewing images individually, then pressing one of navigation buttons causes the currently displayed image to be replaced by the next image.

A digital camera has no film and, as such, there is no incremental cost of taking and storing pictures. Hence, it is possible to take an unlimited number of pictures, wherein the most recent picture replaces the earliest picture, for virtually zero incremental cost. Accordingly, this advantage is best realized when the camera is used as much as possible, taking pictures of practically anything of interest.

One way to best utilize this unique attribute is to make the digital camera and its internally stored images remotely accessible. If the pictures are remotely accessible, the camera could be set to continuously take pictures of scenes and items of interest. Ideally, a user would be able to access those pictures at any time. The user would be able to use a widely available communications medium to access the camera from virtually an unlimited number of locations.

The emergence of the Internet as a distributed, widely accessible communications medium provides a convenient avenue for implementing remote accessibility. Providing remote accessibility via the Internet leverages the fact that the Internet is becoming familiar to an increasing number of

people. Many users have become accustomed to retrieving information from remotely located systems via the Internet. There are many and varied applications which presently use the Internet to provide remote access or remote connectivity. Internet telephony is one such application, such as Microsoft's NetMeeting and Netscape's CoolTalk.

NetMeeting and CoolTalk are both real-time desktop audio conferencing and data collaboration software applications specifically designed to use the Internet as their communications medium. Both software applications allow a "local" user to place a "call" to a "remote" user located anywhere in the world. With both NetMeeting and CoolTalk, the software application is hosted on a personal computer system at the user's location and on a personal computer system at the remote user's location. Both NetMeeting and CoolTalk require a SLIP (Serial Line Internet Protocol) or PPP (Point-to-Point Protocol) account where Internet access is via a dial-up modem, and where the user, as is typical, accesses the Internet through an ISP (Internet Service Provider). Both NetMeeting and CoolTalk require personal computer systems for the resources necessary to run these applications (e.g., processing power, memory, communications hardware, and the like). In addition, both NetMeeting and CoolTalk require the one user to input an IP (Internet Protocol) address for the other user in order to establish communication between the users.

To facilitate the process of obtaining appropriate Internet addresses, CoolTalk, for example, allows on-line users to list their respective IP addresses with a proprietary CoolTalk central Web server. This allows a user to obtain a list of users currently on-line to whom communication can be established. Upon locating the desired remote user in the Internet address list maintained by the Web server, the local user places the call.

In this manner, the proprietary CoolTalk Web server maintains a user-viewable and user-updated "address book" in which users list their respective Internet addresses and in which they search for the Internet addresses of others with whom they wish to communicate. However, both NetMeeting and CoolTalk require active user input, in that each require the user to input his current Internet address, and in that each require the local user to search the address book for the Internet address of the remote user to be contacted. This can be quite problematic in the case where users obtain access to the Internet via dial-up connections and hence have different Internet addresses each time their respective dial-up connections are established.

In a manner similar to Internet telephony, Internet desktop video conferencing is another prior art application which uses the Internet as its communications medium. One such application, for example, is CU-SeeMe by White Pine. CU-SeeMe provides real time video conferencing between two or more users. As with NetMeeting and CoolTalk, CU-SeeMe is a software application which runs on both the local user's personal computer system and the remote user's personal computer system. The personal computer systems provide the resources for running the application. As with NetMeeting and CoolTalk, CU-SeeMe requires the local user to enter the IP address of the remote user. Like CoolTalk, CU-SeeMe facilitates this process by allowing on-line users to list their respective IP addresses with a proprietary central Web server such that the addresses can be easily indexed and searched.

Another prior art example of remote access via the Internet is status queries of remote devices using the Internet as the communications medium. A typical prior art applica-

3

tion involves interfacing a remote device with a computer system, and accessing the computer system via the Internet. For example, a vending machine can be remotely accessed to determine its status (e.g., the number of sales made, whether the machine needs refills, whether the machine needs maintenance, and the like). The machine is appropriately equipped with sensors, switches, and the like, which in turn are interfaced to a computer system using a software driver. The computer system is coupled to the Internet and interfaces with the machine through the driver, making the relevant information available over the Internet using Web server software. Hence, any interested user (e.g., the vending machine service company) is able to remotely ascertain the status of the machine via the Internet.

A problem with the above described prior art applications is that access to the Internet and communication thereon require a separate host computer system (e.g., a personal computer system) on each side of the Internet connection in addition to the server computer system on the Internet. The two host computer systems provide the computational resources to host the respective software applications, the Internet access software, and any necessary device drivers. The required computational resources consume a significant amount of memory. Because of this, among other reasons, the above prior art applications are not easily transferred to the realm of easy-to-use, intuitive, consumer electronic devices such as digital cameras, which are small in size and so generally constrained by the amount of memory they can house. In addition, a consumer electronic device such as a digital camera that requires a separate computer system would be more expensive and complex, and therefore would not be consistent with the desire of consumers for lower cost and simpler devices.

Also, separate host computer systems (where the host computer systems host the software and drivers required by prior art applications as described above) require extra effort to administer, particularly with regard to networks consisting of a large number of computer systems (e.g., digital cameras each incorporating a computer system). For example, an upgrade to the software residing on each computer system has to be individually installed on each computer system. Also, each computer system has to be individually polled to query whether the computer system has data of interest to the user, and then the data have to be separately accessed and collected from each computer system, then compiled. For example, in an application involving digital cameras, a user may be interested in finding out which digital cameras have images in storage. In the prior art, the user has to access each digital camera individually. In another case, a user may have an interest in maintaining a record of transactions between all users and all digital cameras. Again, in the prior art this is accomplished by individually accessing each digital camera (or, alternatively, each user's computer system) to collect the data, and then compiling the complete list of transactions.

Another problem with the prior art is the fact that the applications described above require the user to know the Internet address of the person or device that is being contacted. The Internet telephony applications (e.g., CoolTalk) often employ a user-viewable and user-updated address book to facilitate the process of locating and obtaining the correct Internet address; however, they require active user input. This is difficult in the case where users obtain access to the Internet via dial-up connections, and thus have changing Internet addresses. Still another problem with the prior art is that the applications described above provide only a limited degree of functionality; that is, they are

4

limited to either chat, video conferencing, or the like. As such, they are not capable of establishing a connection between any type of user system and remote device.

One prior art system is described in the copending previously filed patent application, assigned to the assignee of the present invention, entitled "A Method and System for Hosting an Internet Web Site on a Digital Camera," Eric C. Anderson and others, Ser. No. 09/044,644. This prior art system presents one solution to the problem of gaining remote access to those digital devices where the location and Internet address of the device are highly changeable. This prior art system incorporates a Web server into the digital device, specifically a digital camera. However, the disadvantage to this prior art system is that the Web server consumes valuable memory and computational resources in the digital camera. In addition, because of the limited memory in a device such as a digital camera, the Web server is not as powerful as a Web server on a server computer system.

Thus, a need exists for an inexpensive and powerful method for implementing remote access to digital devices, such as digital cameras, where the location and Internet address of the device are highly changeable. A further need exists for an intuitive, simple protocol for presenting the device's functionality and capabilities to users. In addition, a need exists for a method of efficiently administering a plurality of separate devices. A need also exists for an efficient process of obtaining the address of the device that is transparent to the viewer. The present invention provides a novel solution to the above needs.

SUMMARY OF THE INVENTION

The present invention provides an inexpensive and powerful method for implementing remote access to digital devices, such as digital cameras, where the location and Internet address of the device are highly changeable. The present invention further provides an intuitive, simple protocol for presenting the device's functionality and capabilities to users. In addition, the present invention provides a method of efficiently administering a plurality of separate devices. The present invention also provides an efficient process of obtaining the address of a device that is transparent to the user.

In one embodiment, the present invention is an executable program for accessing a digital camera via a communication network using a Web server on a server computer system and a Web browser (or a program of similar function) on a client computer system that are communicatively coupled via the Internet. The address of the digital camera is registered in an executable program on the server computer system. The executable program is accessed by the client computer system. The executable program connects the digital camera and the server computer system. The executable program enables the client computer system and the digital camera to communicate using any protocol used by these devices, thus allowing data (e.g., images) acquired by the digital camera to be transferred to the client computer system.

The executable program can be implemented in a variety of forms. For example, the executable program can be a Java script. Alternatively, the executable program can be a cgi-bin (Common Gateway Interface-binaries).

For example, in the case of a digital camera, the executable program directly communicates commands from the client computer system to the digital camera when both the client computer system and the digital camera are on-line at the same time. Alternatively, if the digital camera is not

5

on-line, the commands from the client computer system are first stored in the server computer system, and then later communicated by the executable program to the digital camera after a connection between the server computer system and the digital camera is made. The capability to store and then forward commands and data is not limited to a digital camera application nor is it limited to a particular data storage format. The data storage format can be any format that is understood by both the client computer system and the digital device.

Images and any other data acquired by the digital camera are accessed by the server computer system using the executable program and directly transferred to the client computer system when both the client computer system and the digital camera are on-line at the same time. Alternatively, if the client computer system is not on-line, the data are first stored by the server computer system, and then later communicated to the client computer system after a connection between the server computer system and the client computer system is made.

It should be noted, however, that the present invention can be readily modified to function in other embodiments, such as, for example, hand-held digital devices, lap top personal computers, and the like, which require an efficient process of obtaining the address of a device that is transparent to the user.

These and other objects and advantages of the present invention will become obvious to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments which are illustrated in the various drawing figures.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not by way of limitation, in the figures of the accompanying drawings in which like reference numerals refer to similar elements and in which:

FIG. 1A shows a block diagram of a remote access system via the Internet in accordance with one embodiment of the present invention.

FIG. 1B shows a block diagram of a remote access system via a Local Area Network in accordance with one embodiment of the present invention.

FIG. 1C shows a block diagram of a digital camera coupled to the Internet via an Internet Service Provider.

FIG. 2 shows a general purpose computer system upon which embodiments of the present invention may be practiced.

FIG. 3 shows a block diagram of a digital camera for use in accordance with the present invention.

FIG. 4 shows a block diagram of a computer system of a digital camera in accordance with one preferred embodiment of the present invention.

FIG. 5 shows a memory map of a dynamic random access memory of a digital camera in accordance with one embodiment of the present invention.

FIG. 6 shows a diagram of the connectivity and application software of a digital camera in accordance with one embodiment of the present invention.

FIG. 7 is a flowchart of a process for remotely accessing a digital camera implemented by an executable program in accordance with one embodiment of the present invention.

FIG. 8 is a flowchart of a process employed by a digital camera for remote access in accordance with one embodiment of the present invention.

6

FIG. 9 is a flowchart of a process employed by a client computer system for remote access of a digital camera in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In the following detailed description of the present invention, numerous specific details are set forth in order to enable one of ordinary skill in the art to make and use the invention, and are provided in the context of a patent application and its requirements. Although the present invention will be described in the context of a digital camera, various modifications to the preferred embodiment will be readily apparent to those skilled in the art, and the generic principles herein may be applied to other embodiments. That is, any digital device which displays data, images, icons and/or other items, could incorporate the features described below and that device would be within the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

Some portions of the detailed descriptions which follow are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computer memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is here, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computer system. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, bytes, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that throughout the present invention, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes (e.g., the processes of FIGS. 7, 8 and 9) of a computer system or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present invention provides a method for making a digital device (e.g., a digital camera) and its internally stored data remotely accessible over a communication network such as the Internet or a Local Area Network (LAN). The present invention is an executable program placed on a server computer system (specifically, a Web server) that implements commands initiated by a client (or user) using a client computer system with a Web browser or a program of similar function. The present invention enables the digital camera to be set to continuously take pictures of scenes/

7

items of interest and allow a client to access those pictures at any time. The present invention allows the client to access the digital camera from virtually an unlimited number of locations and with the camera in virtually any location.

Referring now to FIG. 1A, a block diagram of communication network 100 is shown. Communication network 100 provides a method in accordance with one embodiment of the present invention which implements remote access to camera 300 and its internally stored data. Communication network 100 includes camera 300, Internet Service Provider (ISP) 110, Internet Service Provider 115, client (or user) computer system 120, and server computer system 190. ISP 110 and ISP 115 are both directly coupled to the Internet 150. Client computer system 120 includes Web browser 121 or a program of similar function, and server computer system 190 includes Web server 161. Web browser 121 interprets HTML (HyperText Mark-up Language) documents and other data retrieved by Web server 161.

In the present embodiment of the present invention, an executable program resides on server computer system 190, specifically on Web server 161. The executable program implements and manages the connection between server computer system 190, client computer system 120, and camera 300. The executable program can be implemented as a Java servlet, as a cgi-bin (Common Gateway Interface-binaries), or as a similar type of application.

With reference still to FIG. 1A, client computer system 120 is communicatively coupled to ISP 115 via a POTS (plain old telephone system) dial-up connection. Client computer system 120 is coupled to the Internet 150 via one of a bank of modems maintained on the premises of ISP 115. ISP 115 is coupled directly to the Internet via an all-digital connection (e.g., a T1 line). However, other means of coupling client computer system 120 to the Internet 150 may be used in accordance with the present invention.

As depicted in FIG. 1A, camera 300 is communicatively coupled to server computer system 190 via the Internet 150 using a dial-up connection to ISP 110 via a POTS line. Digital camera 300 accesses ISP 110 using a modem, coupling to one of a bank of modems maintained on the premises of ISP 110. ISP 110 is in turn coupled directly to the Internet 150 via an all-digital connection. However, other means of coupling camera 300 to the Internet 150 may be used in accordance with the present invention.

With reference still to FIG. 1A, it should be further appreciated that while communication network 100 shows camera 300 coupling to Internet 150 via one ISP (e.g., ISP 110) and user 120 coupling to Internet 150 via a separate ISP (e.g., ISP 115), user 120 and camera 300 could be coupled to Internet 150 through a single ISP. In such a case, user 120 and camera 300 would be coupled to two separate access ports (e.g., two separate modems out of a bank of modems) of the same ISP.

With reference now to FIG. 1B, in another embodiment of the present invention, the communication network is comprised of Local Area Network (LAN) 170. For example, LAN 170 may be a communication network located within a firewall of an organization or corporation. Client (or user) computer system 171 and server computer system 172 are communicatively coupled via communication line 173a. Client computer system 171 includes an application that is analogous to a Web browser for interpreting HTML documents and other data. Similarly, server computer system 172 includes an application analogous to a Web server for retrieving HTML documents and other data. Camera 300 can be coupled to server computer system 172 through any

8

of a variety of means known in the art. For example, camera 300 can be coupled to server computer system 172 via communication line 173b of LAN 170. This coupling can be accomplished over any network protocol that supports a persistent network connection, such as TCP (Transmission Control Protocol), NetBIOS, IPX (Internet Packet Exchange), and LU6.2, and link layers protocols such as Ethernet, token ring, and ATM (Asynchronous Transfer Mode). Alternatively, camera 300 can be coupled to server computer system 172 via an input/output port (e.g., a serial port) of server computer system 172.

Referring now to FIG. 1C, a more detailed diagram of camera 300 coupled to the Internet 150 is shown. Camera 300 is coupled to an external modem 101. Camera 300 is coupled to modem 101 via any of several communications means (e.g., Universal Serial Bus, infrared link, and the like). Modem 101 is in turn coupled to a POTS telephone jack 102 at the camera's location. Telephone jack 102 couples modem 101 to one of modems 103 of ISP 110 via the telephone company's local loop. ISP 110 is directly coupled to the Internet 150 via an all digital connection (e.g., a T1 line).

Continuing with reference to FIG. 1C, modem 101 is shown as an external modem. However, the functionality of modem 101 can be implemented directly within the electronics of camera 300 (e.g., via a modem application-specific integrated circuit or ASIC), or alternatively can be implemented as a software-only modem executing on a computer within camera 300. As such, it should be appreciated that, at the hardware connectivity level, modem 101 can take several forms. For example, a wireless modem can be used in which case the camera is not connected via an external wire to any land line. Alternatively, there may be applications in which camera 300 includes suitable electronic components enabling a connection to a conventional computer system network (e.g., Ethernet, AppleTalk, and the like), which is in turn directly connected to the Internet (e.g., via a gateway, a firewall, and the like), thereby doing away with the requirement for an ISP. Hence, it should be appreciated that the present invention is not limited to any particular method of accessing the Internet 150.

Refer now to FIG. 2 which illustrates server computer system 190 upon which embodiments of the present invention may be practiced (the following discussion is also pertinent to a client computer system). In general, server computer system 190 comprises bus 200 for communicating information, processor 201 coupled with bus 200 for processing information and instructions, random access memory 202 coupled with bus 200 for storing information and instructions for processor 201, read-only memory 203 coupled with bus 200 for storing static information and instructions for processor 201, data storage device 204 such as a magnetic or optical disk and disk drive coupled with bus 200 for storing information and instructions, optional display device 205 coupled to bus 200 for displaying information to the computer user, optional alphanumeric input device 206 including alphanumeric and function keys coupled to bus 200 for communicating information and command selections to processor 201, optional cursor control device 207 coupled to bus 200 for communicating user input information and command selections to processor 201, and network interface card (NIC) 208 coupled to bus 200 for communicating from a communication network to processor 201.

Display device 205 of FIG. 2 utilized with server computer system 190 of the present invention may be a liquid crystal device, cathode ray tube, or other display device

suitable for creating graphic images and alphanumeric characters recognizable to the user. Cursor control device 207 allows the computer user to dynamically signal the two-dimensional movement of a visible symbol (pointer) on a display screen of display device 205. Many implementations of the cursor control device are known in the art including a trackball, mouse, joystick or special keys on alphanumeric input device 206 capable of signaling movement of a given direction or manner of displacement. It is to be appreciated that the cursor control 207 also may be directed and/or activated via input from the keyboard using special keys and key sequence commands. Alternatively, the cursor may be directed and/or activated via input from a number of specially adapted cursor directing devices.

Referring now to FIG. 3, a block diagram of digital camera 300 is shown for use in accordance with the present invention. Camera 300 preferably comprises imaging device 314, system bus 316 and computer 318. Imaging device 314 is optically coupled to object 312 and electrically coupled via system bus 316 to computer 318. Once a photographer has focused imaging device 314 on object 312 and, using a capture button or some other means, instructed camera 300 to capture an image of object 312, computer 318 commands imaging device 314 via system bus 316 to capture raw data representing object 312. The captured raw data are transferred over system bus 316 to computer 318, which performs various processing functions on the data before storing it in memory. System bus 316 also passes various status and control signals between imaging device 314 and computer 318.

Referring now to FIG. 4, a block diagram of one embodiment of computer 318 is shown. System bus 316 provides connection paths between imaging device 314, an optional power manager 442, central processing unit (CPU) 444, dynamic random-access memory (DRAM) 446, input/output interface (I/O) 448, non-volatile memory 450, and buffers/connectors 452. Removable memory 454 connects to system bus 316 via buffers/connectors 452. Alternatively, camera 300 may be implemented without removable memory 454 or buffers/connectors 452.

Power manager 442 communicates via line 466 with power supply 456 and coordinates power management operations for camera 300. CPU 444 typically includes a conventional processor device for controlling the operation of camera 300. In the present embodiment, CPU 444 is capable of concurrently running multiple software routines to control the various processes of camera 300 within a multi-threaded environment. DRAM 446 is a contiguous block of dynamic memory which may be selectively allocated to various storage functions. LCD controller 490 accesses DRAM 446 and transfers processed image data to LCD screen 402 for display.

I/O 448 is an interface device allowing communications to and from computer 318. I/O 448 permits an external device (not shown) to connect to and communicate with computer 318. I/O 448 also interfaces with a plurality of buttons and/or dials 404, and optional status LCD 406, which in addition to LCD screen 402, are the hardware elements of the camera's user interface 408.

Non-volatile memory 450, which may typically comprise a conventional read-only memory or flash memory, stores a set of computer-readable program instructions to control the operation of camera 300.

Referring now to FIG. 5, a memory map showing one embodiment of dynamic random access memory (DRAM) 446 is shown. In the present embodiment, DRAM 446 includes RAM disk 532, system area 534, and working memory 530.

RAM disk 532 is a memory area used for storing raw and compressed data and typically is organized in a "sectored" format similar to that of conventional hard disk drives. In the present embodiment, RAM disk 532 uses a well-known and standardized file system to permit external devices, via I/O 448 of FIG. 4, to readily recognize and access the data stored on RAM disk 532. System area 534 typically stores data regarding system errors (for example, why a system shutdown occurred) for use by CPU 444 (FIG. 4) upon a restart of computer 318 (FIG. 3).

Working memory 530 includes various stacks, data structures and variables used by CPU 444 while executing the software routines used within computer 318. Working memory 530 also includes several input buffers 538 for temporarily storing sets of raw data received from imaging device 314 (FIG. 3), and frame buffer 536 for storing data for display on LCD screen 402 (FIG. 4). In the present embodiment, each input buffer 538 and frame buffer 536 are split into two separate buffers (shown by the dashed lines) to improve the display speed of the digital camera and to prevent the tearing of the image in LCD screen 402.

Referring now to FIG. 6, a diagram of the connectivity and application software of camera 300 of FIG. 3 is shown. At the software level, computer 318 (FIG. 3) of camera 300 hosts any network protocol that supports a persistent network connection. This coupling can be accomplished over any network protocol that supports a persistent network connection, such as TCP/IP (Transmission Control Protocol/Internet Protocol) including Point-to-Point Protocol, NetBIOS, IPX, and LU6.2, and link layers protocols such as Ethernet, token ring, and ATM. Protocol stack 601 interfaces with the communications hardware 602 (e.g., a modem) of camera 300 and the application layer 603. The bottom of protocol stack 601 includes communication hardware interface drivers which interface directly with the various communications hardware with which camera 300 must function (e.g., a Universal Serial Bus and the like). Applications layer 603, protocol stack 601, and operating system 604 are installed as software modules in DRAM 446 (FIG. 4) of camera 300. Software applications within applications layer 603 interface with operating system 604. Operating system 604 controls the hardware functionality of camera 300 (e.g., taking pictures, storing pictures, and the like) via camera hardware 605. Incoming data 620, such as HTTP (Hypertext Transfer Protocol) requests and the like, are received and outgoing data 630, such as HTML (Hypertext Markup Language) files and the like, are transferred to and from camera 300 via protocol stack 601 and communications hardware 602. Web browser 121 of FIG. 1A (or a program of similar function) can process data files, launch plug-ins, and run Java applets that communicate with camera 300 in a variety of methods in addition to those methods involving an exchange of HTML files.

FIG. 7 illustrates an executable program 700 for client computer system 120 (specifically, Web browser 121 of FIG. 1A or a program of similar function) to remotely access camera 300 (FIG. 3), where executable program 700 is implemented in accordance with the present invention as program instructions stored in computer-readable memory units (e.g., read-only memory 203) and implemented by processor 201 of server computer system 190 of FIG. 1A (specifically, by Web server 161). Executable program 700 performs functions both for and in response to Web browser 121 (or a program of similar function) and camera 300. The description below first discusses the steps associated with setting up a connection between executable program 700 and camera 300, then discusses the steps associated with

11

setting up a connection between executable program 700 and Web browser 121 (or a program of similar function), however, the present invention is not limited by the order in which these steps are presented.

In the present embodiment, executable program 700 is identified and accessed by its own unique address, commonly referred to as an URL (Unified Resource Locator), as is well known in the art. The URL for executable program 700 fully describes where it resides on a communication network (e.g., the Internet 150) and how it is accessed. In the present embodiment, included in the URL for executable program 700 is the name of camera 300. Accordingly, in one embodiment using a servlet for executable program 700, a standard format for a URL is: `http://webserver/HostName/cameraServlet/WellKnownName/cameraName`.

In step 705, executable program 700 receives and accepts a connection request from camera 300. Executable program 700 runs constantly on Web server 161 and is configured to listen for connection requests on a plurality of communication protocols (e.g., TCP, NetBIOS, and the like). In the present embodiment, camera 300 is connected to executable program 700 via Web server 161 as described in conjunction with FIG. 1A, and opens the connection using the connectivity and application software described in conjunction with FIG. 6.

In step 710, executable program 700 receives and reads registration information from camera 300. As mentioned above, executable program 700 is configured to communicate using a number of different communication protocols. Such registration information includes the name of the camera and authentication information such as security information and account information. Executable program 700 uses the camera name to identify the camera and locate it in response to a client request.

In step 720, executable program 700 compares the registration information with a predefined access control list to determine if camera 300 is a camera for which Web server 161 is to provide support and service. If not, executable program 700 closes the connection between Web server 161 and camera 300.

In step 725, upon successful completion of step 720, executable program 700 registers camera 300 and stores the camera's name and associated requirements, such as security and account information. Executable program 700 also sends a message that camera 300 is registered. At this point, the connection between executable program 700 can be either terminated or maintained at the option of the camera's operator. If the connection is terminated, the registration information is maintained by Web server 161 and can be later accessed by executable program 700 when a subsequent connection is made with camera 300.

In step 730, executable program 700 accepts the connection request from camera 300 and thus has a persistent and long term connection with camera 300. As described above, the connection can be an ongoing connection maintained from the time when camera 300 was first registered. New connections can be made in the future whenever camera 300 reinitiates the registration protocol. Once camera 300 and executable program 700 have established a connection, they then wait until a client also makes a connection to access the camera. However, as will be described below in conjunction with step 770, it is not necessary for there to be an open connection between the Web server and the camera at the same time that there is an open connection between the Web browser (or a program of similar function) and Web server to accomplish remote access of the camera in accordance with the present invention.

12

In step 740, executable program 700 receives and accepts a request for a connection from a client. The client enters the URL of executable program 700, including the name of the camera (e.g., camera 300) to which access is desired, into a Web browser (e.g., Web browser 121 on client computer system 120 of FIG. 1A or a program of similar function). Using standard communication protocols such as TCP/IP, Web server 161 is queried with the URL for executable program 700. Web server 161 recognizes the URL and makes the connection to executable program 700. Thus, the present invention establishes a single location identified by a known URL where the client always goes to connect to the camera, no matter where the camera is or where the client is. (For those cases in which TCP/IP is not available—for example, when the device is not attached to the Internet or the like—camera 300 connects to executable program 700 directly using a well-known address that is associated with the protocol being used, such as a NetBIOS name.)

In step 745, executable program 700 then assumes control over the TCP/IP connection between Web browser 121 (or a program of similar function) and Web server 161. Executable program 700 establishes a persistent and long term connection between the browser and server. That is, the connection between Web browser 121 (or a program of similar function) and Web server 161 is kept open by executable program 700. As will be described below in conjunction with step 770, it is not necessary for there to be an open connection between the client computer system and the Web server at the same time that there is an open connection between the Web server and the camera to accomplish remote access of the camera in accordance with the present invention.

In step 750, executable program 700 next determines if camera 300 is registered as discussed in conjunction with step 735. If camera 300 is not registered, executable program 700 sends an appropriate message to the client to indicate such.

In step 760, if the camera is registered, executable program 700 validates the required access information provided by the client against the security and account information provided when camera 300 was registered. For example, executable program 700 validates whether the client is utilizing an authorized password or user name. If not, executable program 700 transmits an appropriate message to the client. The present invention can optionally provide additional services related to security or account information. For example, it could control the type of access a client is permitted based on the authentication information received from the client, or it could verify credit information and bill the client for services requiring payment.

In step 765, upon satisfactory completion of step 760, executable program 700 extends the connection from Web browser 121 (or a program of similar function) to camera 300 if there is an established connection to the camera as described in conjunction with step 730. Hence, a client using Web browser 121 or a program of similar function has direct, remote access to camera 300 via executable program 700 in Web server 161.

In step 770, executable program 700 forwards commands from a client to camera 300, and forwards images and other data from camera 300 to the client via Web server 161 and Web browser 121 (or a program of similar function). That is, executable program 700 enables a direct communication between the client computer system and the camera allowing the client to remotely access and manage the camera. If the client and the camera are both concurrently connected to

13

executable program 700, then the client immediately receives the data, and camera 300 immediately executes any commands from the client.

However, if Web browser 121 (or a program of similar function) and camera 300 are not each connected at the same time, remote access to the camera is still accomplished in accordance with the present invention. If camera 300 is not on-line, a client uses Web browser 121 (or a program of similar function) to access Web server 161 and executable program 700. The client transmits commands, and executable program 700 stores the commands on Web server 161. The client may then close the connection to the Web server. Subsequently, when camera 300 opens a connection to Web server 161 and executable program 700, executable program 700 retrieves the commands and forwards them to the camera. Camera 300 downloads the commands and executes them. The results from executing the commands are then sent to executable program 700, which stores them on Web server 161 until they are retrieved by the client.

Similarly, if camera 300 establishes a connection with Web server 161 and executable program 700 but the client is not on-line, the camera can, for example, download images and any other data that executable program 700 stores on Web server 161. Camera 300 may then close the connection to the Web server. The client then later makes a connection to Web server 161 and executable program 700, which retrieves the data and forwards it to the client. The client can also enter commands at this time, which are stored by executable program 700 as described above.

In summary, with references to FIG. 1A and FIG. 7, the client, via Web browser 121 or a program of similar function, Web server 161 and executable program 700, accesses camera 300 to request and retrieve data. Web browser 121 or a program of similar function inserts the Internet address inside the data request (e.g., a HTTP request) and sends the request to Web server 161. Web server 161 receives the data request and associates the request with executable program 700, which in turn assumes the connection between Web server 161 and Web browser 121 (or a program of similar function), and which also establishes a connection between Web browser 121 (or a program of similar function) and camera 300. Executable program 700 subsequently forwards commands from the client to camera 300, and retrieves the requested data (e.g., a HTML file) containing the data and sends it back to Web browser 121 (or a program of similar function). Web browser 121 (or a program of similar function) then interprets the commands and displays the resulting image. The process of accessing a data file from a Web server is commonly referred to as accessing a Web page. Similarly, the process of sending data files from a Web server to a Web browser is commonly referred to as sending a Web page.

Thus, as described above, the present invention provides an intuitive and easy-to-use interface enabling remote access between a client and a camera. By functioning with widely used and familiar Web browsers (or programs of similar function) using standard format URLs to identify the executable program and camera, the present invention provides a simple and familiar interface for accessing the camera. By registering the camera and using an unchanging URL, name to identify the executable program and the camera, the present invention enables the client to locate and access the camera from any remote location no matter where the camera is located. In addition, by using a Java servlet or a cgi-bin for the executable program, the present invention is supported by commonly used Web servers and is readily implemented.

14

Executable program 700 is located on the Web server and not on camera 300, so it does not require additional and substantial memory dedicated to enabling remote access. As such, the present invention permits remote access within the constraints of the size of the camera. In addition, in accordance with the present invention, camera 300 does not require a separate, external computer system (e.g., a personal computer system) for connecting to ISP 110 (FIG. 1A) or for implementing commands and transmitting data, thus providing an inexpensive method for providing remote access to cameras.

Also, by locating the present invention on a Web server (e.g., Web server 161 of FIG. 1A), the Web server becomes a focal point for accessing and managing a plurality of cameras that otherwise would have to be managed and configured separately. For example, executable program 700 on Web server 161 could be updated with new software, and in effect all cameras accessed through that executable program would automatically be updated as well. As another example, executable program 700 could be configured to compile data regarding interactions between clients and all cameras accessed by the executable program. In another example, in accordance with the present invention, a client needs to go only to a single location to determine which of a plurality of cameras served by the executable program have data that have been downloaded to the Web server. Thus, instead of having to access a number of cameras separately, the present invention establishes a single location from which a client can access information about several cameras.

By functioning with a Web-based interface and widely used Web browsers (or programs of similar function), the present invention provides a simple, intuitive, and familiar interface for accessing camera 300's functionality. Accordingly, camera 300's controls and functions are intuitively easy to utilize. Since Web pages and their associated controls (e.g., push buttons, data entry fields, and the like) are very familiar to most users, the remote access functionality of camera 300 can be utilized without requiring a extensive learning period for new users. For example, a consumer purchasing a remotely accessible camera is typically able to easily and immediately use the remote accessibility functions with minimal set-up.

FIG. 8 illustrates a process 800 for remotely accessing camera 300 (FIG. 3), where process 800 is implemented as program instructions stored in computer-readable memory units (e.g., non-volatile memory 450 of FIG. 4) and implemented by CPU 444 (FIG. 4) of camera 300 in accordance with the present invention. FIG. 8 illustrates the process for remotely accessing a camera from the perspective of the actions performed by the camera and the camera operator in accordance with the present invention.

In step 805, camera 300 of FIG. 3 opens a connection to communication network 100 of FIG. 1A. In the present embodiment, camera 300 is coupled to server computer system 190 (specifically, Web server 161) as described in conjunction with FIG. 1A, and opens the connection using the connectivity and application software described in conjunction with FIG. 6. As described in conjunction with FIG. 1C, in the present embodiment, camera 300 couples directly to the telephone system such that a separate and dedicated computer system (e.g., a personal computer system) is not necessary.

In step 810, with the connection made to Web server 161, camera 300 requests a connection to executable program 700 (FIG. 7). For cases in which camera 300 is accessing

15

executable program 700 via, for example, a TCP/IP network such as the Internet 150, then executable program 700 is identified with a URL that is used by the camera to access the executable program. For those cases in which TCP/IP is not available (e.g., when the device is not attached to the Internet or the like), camera 300 connects to executable program 700 directly using a well-known address that is associated with the protocol being used, such as a NetBIOS name.

In step 820, with the camera connected to the executable program, camera 300 registers with executable program 700. For example, camera 300 provides information including an identification name and authentication information such as a password and account information. The information is electronically transmitted from camera 300 and read by executable program 700. Based on this information, the connection between camera 300 and executable program 700 is established if the camera is of the type that is designated to be supported by Web server 161.

In step 830, camera 300 and executable program 700 are linked via a persistent and long term connection; that is, the connection remains open awaiting a client to request access to the camera via Web server 161. As discussed above in conjunction with FIG. 7, it is not necessary for the camera and a client to be connected at the same time to executable program 700.

In step 835, camera 300 receives commands from and transmits data to a client using a Web browser or a program of similar function on a client computer system (e.g., client computer system 120 and Web browser 121 of FIG. 1A or a program of similar function). As described above in conjunction with FIG. 7, the commands and data can be transmitted through an active connection or stored on the Web server.

In the present embodiment, camera 300 is provided with several different operating modes for supporting various camera functions. In capture mode, camera 300 supports the actions of preparing to capture an image and of capturing an image. In review mode, camera 300 supports the actions of reviewing camera contents, editing and sorting images, and printing and transferring images. In play mode, camera 300 allows the client to view screen-sized images in the orientation that the image was captured. Play mode also allows the client to hear recorded sound associated to a displayed image, and to play back sequential groupings of images, which may comprise time lapse, slide show, and burst image images. The client preferably switches between the capture, review, and play modes.

Camera 300 is capable of implementing a wide variety of remote access and remote imaging/surveillance applications. In the present embodiment, camera 300 only records successive images for remote access by the client. The images are loaded into the camera's memory on a first-in, first-out (FIFO) basis, with the earliest recorded image being replaced by the latest recorded image. The number of images available to the client depends upon the amount of installed memory in the camera.

With reference still to step 835 of FIG. 8, when the client and the camera are connected to the Web server at the same time, the commands can be transmitted directly from Web browser 121 (or a program of similar function) to camera 300 via executable program 700 on Web server 161, and camera 300 then executes the commands on-line. Alternatively, when camera 300 is not connected to executable program 700, a client can store commands on Web server 161 by accessing the Web server in a normal fashion,

16

then entering the commands to be stored via executable program 700. Camera 300 then executes the commands when it subsequently connects with executable program 700.

Also in step 835, camera 300 transmits data to a client. Similar to the above, data from camera 300 can be transmitted directly to Web browser 121 or a program of similar function via executable program 700 on Web server 161, when the client and the camera are connected to the Web server at the same time. Alternatively, when a client is not connected to executable program 700, the data are stored on Web server 161 by executable program 700, which then retrieves and transmits the data to a client when the client subsequently connects with executable program 700.

This process of accessing camera 300 from Web browser 121 or a program of similar function occurs transparently with respect to the client. In a typical case, for example, the client types the URL for executable program 700 (which includes the name of camera 300) into Web browser 121 (or a program of similar function) and his "enter" or "return". In accordance with the present invention, the next Web page the client views is the image generated by the data returned from camera 300. Beyond entering the URL for executable program 700 including camera 300, no further action from the client is required in order to access the Web pages hosted by camera 300.

FIG. 9 illustrates a process 900 for remotely accessing camera 300 (FIG. 3), where process 900 is implemented as program instructions stored in computer-readable memory units (e.g., read-only memory) and implemented by the central processor of client computer system 120 (specifically, Web browser 121 or a program of similar function) of FIG. 1A. FIG. 9 illustrates the process for remotely accessing a camera from the perspective of the actions performed by the client in accordance with the present invention.

In step 905, the client opens a network connection such as a dial-up connection to ISP 115 of FIG. 1A.

In step 910, the client enters into a Web browser (e.g., Web browser 121 on client computer system 120 of FIG. 1A or a program of similar function) the URL of executable program 700, including the name of the camera (e.g., camera 300) to which access is desired. Alternatively, the client enters the URL of executable program 700 only. (For those cases in which TCP/IP is not available—for example, when the device is not attached to the Internet or the like—camera 300 connects to executable program 700 directly using a well-known address that is associated with the protocol being used, such as a NetBIOS name.) Using standard communication protocols such as TCP/IP, Web server 161 is queried with the URL for executable program 700. Web server 161 recognizes the URL and makes the connection to executable program 700. A persistent and long term two-way connection is opened between the client on Web browser 121 (or a program of similar function) and executable program 700.

In step 915, the client enters the authentication information required to gain access to executable program 700 and camera 300.

In step 920, as explained above, from his/her remote location the client sends commands to camera 300 and receives images from the camera. As explained above, the client and the camera do not have to be connected to the Web server at the same time to enable remote access. Depending on the particular application, the Web page for camera 300 can include control buttons, data entry fields, drop-down menus, and the like.

17

Thus, executable program 700 enables the client to access from a remote location the functional controls of camera 300 as well as the images and other data acquired by the camera.

Pseudo-Code Sections A, B, C, D and E below provide additional details regarding processes 700, 800 and 900 of FIGS. 7, 8 and 9. Pseudo-Codes Sections A through E represent the method of one embodiment of the present invention. However, it is appreciated that other embodiments are possible in accordance with the present invention.

18

With references to FIGS. 7 and 9, the pseudo-code for the connection of a client to the server computer system, specifically to executable program 700, is described in accordance with one embodiment of the present invention in Pseudo-Code Section A below.

Pseudo-Code Section A: Client Connection Setup

```

1. Client sends HTTP Post command to http://serverName/gateway device="deviceName"
   Note: authentication/security data is optional and is handled by the browser.
2. Client receives HTTP response from Gateway
3. If response is OK or OKDataPresent
{
    Client has TCP connection it can use to send/receive anything to/from the device
    The protocol used can be any that the Client & Device agree upon.
}
else
{
    The response is failed or DataPresent.
    The TCP connection is closed:
}

```

Note: the OKDataPresent and DataPresent responses are informational. The Client at any time on separate connections may issue a CacheData or GetCache command to the Gateway. See algorithms C.

30 With references to FIGS. 7 and 8, the pseudo-code for the connection of a device (e.g., digital camera 300 of FIG. 1A) to the server computer system, specifically to executable program 700, is described in accordance with one embodiment of the present invention in Pseudo-Code Section B below.

Pseudo-Code Section B: Device Connection Setup/Registration

```

1. Device establishes a connection to the Gateway using any networking protocol
   supported by the Gateway (TCP, NetBIOS, IPX, 802.2, etc) using a known address.
2. Device sends a Register request to the Gateway on the connection passing
   its name (and optionally authentication/security information).
3. Device receives response from Gateway
4. If the response is OK or OKDataPresent
{
    The Device may use the existing connection to wait for requests from clients
    or the Device may optionally close the connection.
}
else
{
    The response is Failed or DataPresent
    The connection is closed.
}

```

Note: the OKDataPresent and DataPresent responses are informational. The Device at any time on separate connections may issue a CacheData or GetCache command to the Gateway. See algorithm D.

60

With reference to FIG. 7, the pseudo-code for executable program 700 in response to a client connection request is described in accordance with one embodiment of the present invention in Pseudo-Code Section C below.

65 Pseudo-Code Section C: Gateway Handles Client Connection Request

```

Start:
    Wait for incoming requests from clients
    if its an HTTP Post command with parameters CacheData and deviceName
    {
        go to CacheData
    }
    else if its an HTTP Get command with parameters GetCache and deviceName
    {
        go to GetCache
    }
    if its an HTTP Post command from a client with the name of device
    {
        go to Connect
    }
    else
    {
        go to Fail
    }
Connect:
    Look up device by name in registry
    if the device entry is found
    {
        if there is a device connection id in the entry
        {
            if the entry is not busy
                go to Success
            else go to Fail
        }
        else
        {
            Attempt to establish connection with device at last known address
            if connection established
            {
                go to Success
            }
            else go to Fail
        }
    }
    else
    {
        send Fail response
        close connection
        go to Start
    }
Fail:
    if there is data in the cache
    {
        set response to DataPresent
    }
    else
    {
        set response to Fail
    }
    send response
    close connection
    go to Start
Success:
    if there is data in the cache
    {
        set response to OkDataPresent
    }
    else
    {
        set response to Ok
    }
    store connection id of the client in the registry entry
    go to Start
CacheData:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        store data in Cache for device with identification information

```


-continued

```

    send Ok response
    close connection
    go to Start;
}
GetCache:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        send data in Cache for this client
        close connection
        go to Start
    }
}

```

With reference to FIG. 7, the pseudo-code for executable program 700 in response to a device connection request is described in accordance with one embodiment of the present invention in Pseudo-Code Section D below.

Pseudo-Code Section D: Gateway Handles Device Requests

```

Start:
    Wait for incoming requests from devices
    Receive incoming request
    if its a Register request
    {
        go to Register
    }
    else if its a CacheData request
    {
        go to CacheData
    }
    else if its a GetCache request:
    {
        go to GetCache
    }
    else
    {
        return Failed
        close connection
        go to Start
    }
Register:
    Note: The gateway may optionally authenticate device
    Get device name from Register request
    look up name in table
    if name is found
    {
        go to Success
    }
    else
    {
        The gateway may optionally add the name or reject the
        registration attempt, then go to Success or Fail
        respectively
    }
Success:
    check for the presence of cached client data
    if cached data is present
    {
        set response to OkDataPresent;
    }
    else
    {
        set response to Ok
    }
}

```

-continued

```

    Store a connection id for the incoming connection in
    the registry entry for the device.
    send response
    go to Start
Fail:
    if cached data from clients is present for this device
    {
        set response to DataPresent
    }
    else
    {
        set response to Fail
    }
    send response
    close connection
    go to Start
CacheData:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        store data in Cache for client
        send Ok response
        close connection
        go to Start;
    }
GetCache:
    Look up device name in registry
    if not found
    {
        set response to Fail
        send response
        close connection
        go to Start
    }
    else
    {
        send data in Cache for device to the device
        close connection
        go to Start
    }
}

```

With reference to FIG. 7, the pseudo-code for executable program 700, specifically the handling of data when an open connection is present between the device and the client

23

computer system, is described in accordance with one embodiment of the present invention in Pseudo-Code Section E below.

Pseudo-Code Section E: Gateway Handles Data on Existing Connections

```

Start:
  Wait for data
  Map the incoming connection id to its partner connection id
  if there is incoming data
  {
    Send the data out on the partner connection id
    go to Start
  }
  else if the connection closed
  {
    if it is a client connection
    {
      Remove the client's connection id from the
      device's entry in the registry.
    }
    else
    {
      Remove both the client and device connection ids
      from the device's entry in the registry.
      Close the client's connection
      go to Start
    }
  }
  go to Start
  
```

As described above, the remote accessibility of camera **300** provides for many new applications of digital imagery. One such application involves setting up camera **300** at some remote location and using it to take pictures at successive intervals. These pictures would be accessed via the Internet **150** as they are taken. The interval can be adjusted (e.g., more or less pictures per minute) in response to commands entered by a client via a Web browser (e.g., Web browser **121** of FIG. 1A or a program of similar function).

Another application involves using camera **300** in conjunction with a motion detector. When used in conjunction with a motion detector, camera **300** can be configured to capture an image in response to receiving a signal from the motion detector (e.g., detecting the motion of an intruder), thereby taking a picture of whatever triggered the detector's signal output. Alternatively, camera **300** can detect motion by simply comparing successive images to detect changes between them, thereby dispensing with the need for a separate motion detector.

Yet another application involves using camera **300** in conjunction with a remote aiming device. Camera **300** can be mounted on a remotely operated aiming device (e.g., a motorized tripod). The aiming device is controlled via the Internet **150** in the same manner the camera is controlled via the Internet **150**. Alternatively, camera **300** could be coupled to control the remote aiming device directly. The remote aiming device allows a client to control the field of view of the camera **300** in the same manner the client controls other functionality (e.g., picture resolution, picture interval, and the like).

In this manner, executable program **700** of the present invention is able to implement sophisticated remote surveillance of the type previously performed by expensive, prior art closed circuit television devices. Unlike the prior art, however, executable program **700** is inexpensive and relatively simple to implement.

Thus, the present invention provides a method for making a digital camera and its internally stored data remotely

24

accessible. The present invention enables the digital camera to be set to continuously take pictures of scenes and items of interest and to allow a user to access those pictures at any time. The present invention implements remote accessibility via a communication network such as the Internet, thus allowing the user to access the digital camera from virtually an unlimited number of locations and with the camera in virtually any location.

A digital camera in accordance with the present invention does not require a separate, external computer system (e.g., a personal computer system) for Internet connectivity, thus providing an inexpensive method for making remotely accessible digital cameras widely available. In addition, a digital camera in accordance with the present invention is accessed via the widely used and very familiar Web browser (or a program of similar function). By functioning with typical, widely used Web browsers (or programs of similar function), the present invention provides a simple, intuitive, and familiar interface for remotely accessing the digital camera's functionality and capabilities. In so doing, the controls and functions of the digital camera are intuitively easy to utilize, and do not require an extensive learning period for new users. The present invention also provides an efficient and user-transparent process of obtaining the address of a digital camera. Also, the present invention provides a method for efficiently administering a plurality of separate digital cameras.

Although the present invention is described in the context of a digital camera, it is not limited to this embodiment. Hence, the present invention does not provide only a limited degree of functionality as in the prior art applications; that is, it is not limited to either chat or video conferencing, or the like. As such, the present invention is capable of establishing a connection between any type of client system and remote device.

The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.

What is claimed is:

1. A method for allowing a client computer to remotely access a digital image capture unit via a communication network wherein a server computer system and said client computer system are communicatively coupled with communication equipment, said method comprising the steps of:
 - a) allowing said digital image capture unit to establish communication with said server computer system over said network, wherein said digital image capture unit includes connectivity software that enables said digital image capture unit to establish a network connection with said server computer system;
 - b) receiving an address of said digital image capture unit and registering said address in an executable program on said server computer system, wherein said digital image capture unit automatically registers said address with said server computer system;
 - c) allowing said client computer system to access said executable program on said server computer system; and

25

- d) establishing direct communication between said client computer system and said digital image capture unit by communicating commands between said client computer system and said digital image capture unit via said executable program, such that data captured by said digital image capture unit is transferred to said client computer system via said server computer system.
2. The method of claim 1 wherein said communication network is the Internet.
3. The method of claim 1 wherein said communication network is a Local Area Network.
4. The method of claim 1 wherein said image capture unit is a digital camera.
5. The method of claim 1 wherein step a) further comprises communicating authentication information between said digital image capture unit and said executable program.
6. The method of claim 1 wherein said executable program is a Java servlet.
7. The method of claim 1 wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).
8. The method of claim 1 wherein step c) further comprises connecting said digital image capture unit to said server computer system via a Local Area Network.
9. The method of claim 1 wherein step c) further comprises connecting said digital image capture unit to said server computer system via an input/output port of said server computer system.
10. The method of claim 1 wherein step c) further comprises connecting said digital image capture unit to said server computer system via the Internet.
11. The method of claim 1 wherein step d) further comprises storing said commands in a memory unit of said server computer system and communicating said commands to said digital image capture unit at a time when a connection is made between said server computer system and said digital image capture unit.
12. The method of claim 1 further comprising the steps of:
- e) accessing via said executable program data acquired by said digital image capture unit; and
 - f) transferring said data from said digital image capture unit to said client computer system via said server computer system.
13. The method of claim 12 further comprising storing said data in a memory unit of said server computer system and communicating said data to said client computer system at a time when a connection is made between said server computer system and said client computer system.
14. A computer system comprising:
- a) a processor coupled to a bus; and
 - a memory unit coupled to said bus and having stored therein an executable program that when executed by said processor implements a method for allowing a client computer to remotely access a digital image capture unit via a communication network, said method comprising the steps of:
 - a) allowing said digital image capture unit to establish communication with said server computer system over said network, wherein said digital image capture unit includes connectivity software that enables said digital image capture unit to establish a network connection with said server computer system;
 - b) receiving an address of said digital image capture unit and registering said address with said executable program, wherein said digital image capture unit automatically registers said address with said server computer system;

26

- c) allowing said client computer system to access said executable program over said communication network; and
 - d) establishing direct communication between said client computer system and said digital image capture unit by communicating commands between said client computer system and said digital capture unit via said executable program.
15. The computer system of claim 14 wherein said computer system is a server computer system.
16. The computer system of claim 14 wherein said digital image capture unit is a digital camera.
17. The computer system of claim 14 wherein said executable program is a Java servlet.
18. The computer system of claim 14 wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).
19. The computer system of claim 14 wherein said digital image capture unit and said computer system are connected via a Local Area Network.
20. The computer system of claim 14 wherein said digital image capture unit and said computer system are connected via an input/output port of said computer system.
21. The computer system of claim 14 wherein said digital image capture unit and said computer system are connected via the Internet.
22. In a communication network comprising a server computer system and a client computer system communicatively coupled with communication equipment, a method for allowing said client computer to remotely access a digital camera via said communication network, said method comprising the steps of:
- a) allowing said digital camera to establish communication with said server computer system over said communication network, wherein said digital camera includes connectivity software that enables said digital camera to establish a network connection with said server computer system;
 - b) receiving an address of said digital camera and registering said address in an executable program on said server computer system, wherein said digital camera automatically registers said address with said server computer system;
 - c) allowing said client computer system to access said executable program on said server computer system;
 - d) establishing direct communication between said client computer system and said digital camera by communicating commands between said client computer system and said digital camera via said executable program;
 - e) accessing via said server computer system data acquired by said digital camera; and
 - f) transferring said data from said digital camera to said client computer system via said server computer system.
23. The method of claim 22 wherein said executable program is a Java servlet.
24. The method of claim 22 wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).
25. A method for allowing a client computer to remotely access a digital image capture device via a communication network wherein a server computer system and said client computer system are communicatively coupled with communication equipment, said method comprising the steps of:
- a) allowing said digital image capture device to establish communication with said server computer system over

27

said communication network, wherein said digital image capture device includes connectivity software that enables said digital image capture device to establish a network connection with said server computer system;

- b) receiving an address of said digital image capture device and registering said address in an executable program on said server computer system, wherein said digital image capture device automatically registers said address with said server computer system;
- c) allowing said client computer system to access said executable program on said server computer system; and

28

- d) establishing direct communication between said client computer system and said digital image capture device by communicating commands between said client computer system and said digital image capture device via said executable program, such that data from said digital image capture device is transferred to said client computer system via said server computer system.

26. The method of claim **25** wherein said executable program is a Java servlet.

27. The method of claim **25** wherein said executable program is a cgi-bin (Common Gateway Interface-binaries).

* * * * *



US00662218B2

(12) **United States Patent**
Mighdoll et al.

(10) Patent No.: **US 6,662,218 B2**
(45) Date of Patent: ***Dec. 9, 2003**

(54) **METHOD OF TRANSCODING DOCUMENTS
IN A NETWORK ENVIRONMENT USING A
PROXY SERVER**

(75) Inventors: **Lee S. Mighdoll**, San Francisco, CA
(US); **Bruce A. Leak**, Palo Alto, CA
(US); **Stephen C. Perlman**, Mountain
View, CA (US); **Phillip Y. Goldman**,
Los Altos, CA (US)

(73) Assignee: **WebTV Networks, Inc.**, Mountain
View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 0 days.

This patent is subject to a terminal dis-
claimer.

(21) Appl. No.: **10/247,885**

(22) Filed: **Sep. 20, 2002**

(65) **Priority Publication Data**

US 2003/0014499 A1 Jan. 16, 2003

Related U.S. Application Data

(63) Continuation of application No. 09/343,067, filed on Jun.
29, 1999, now abandoned, which is a continuation of application
No. 08/656,924, filed on Jun. 3, 1996, now Pat. No.
5,918,013.

(51) **Int. Cl.** **G06F 15/16**

(52) **U.S. Cl.** **709/219; 709/203; 709/217;
709/228; 709/229; 709/246**

(58) **Field of Search** **709/200-203,
709/217-219, 227-229, 231-232, 246-247**

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,575,579 A 3/1986 Simon et al. 178/4
4,852,151 A 7/1989 Dittakavi et al. 379/97

4,922,523 A 5/1990 Hashimoto 379/96
4,975,944 A 12/1990 Cho 379/209
4,995,074 A 2/1991 Goldnan et al. 379/97
5,005,011 A 4/1991 Perlman et al. 340/728
5,095,494 A 3/1992 Takahashi et al. 375/10
5,220,420 A 6/1993 Hoarty et al. 358/86
5,241,587 A 8/1993 Horton et al. 379/92
5,263,084 A 11/1993 Chaput et al. 379/215
5,287,401 A 2/1994 Lin 379/98
5,299,307 A 3/1994 Young 395/161

(List continued on next page.)

OTHER PUBLICATIONS

Rosoff, Matt, Review: "Gateway Destination PC," c/net
inc., 2 pages, Feb. 19, 1996.

Seidman, Robert, Article: "What Larry and Lou Know (That
You Don't)," c/net inc., 2 pages, Jan. 29, 1996.

Stellin, Susan, Article: "The \$500 Web Box: Less is More?"
c/net inc., 2 pages, 1996.

(List continued on next page.)

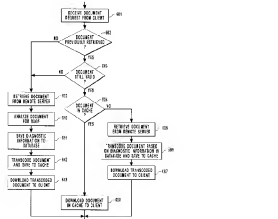
Primary Examiner—Bharat Barot

(74) Attorney, Agent, or Firm—Workman, Nydegger

(57) **ABSTRACT**

A method is described of providing a document to a client
coupled to a server. The server functions as a proxy on
behalf of the client for purposes of accessing a remote
server. In the method, a document is retrieved from the
remote server in response to a request from the client. The
document includes data to be used by the client in generating
a display. The proxying server alters (i.e., transcodes) the
data in the document to form a transcoded document. The
transcoded document is then transmitted to the client. The
proxying server transcodes the data in the document in order
to perform at least one of the following functions: (1)
matching decompression requirements at the client; (2)
converting the document into a format compatible for the
client; (3) reducing latency experienced by the client; and
(4) altering the document to fit into smaller memory space.

31 Claims, 12 Drawing Sheets



U.S. PATENT DOCUMENTS

5,325,423 A	6/1994	Lewis	379/60
5,329,619 A	7/1994	Page et al.	395/200
5,341,293 A	8/1994	Verletney et al.	364/419.17
5,369,688 A	11/1994	Tsukamoto et al.	379/100
5,469,540 A	11/1995	Powers, III et al.	395/158
5,488,411 A	1/1996	Fewis	348/8
5,490,208 A	2/1996	Remillard	379/96
5,530,852 A	6/1996	Meske, Jr. et al.	395/600
5,538,255 A	7/1996	Barker	463/41
5,558,339 A	9/1996	Periman	463/42
5,561,709 A	10/1996	Remillard	379/96
5,564,001 A	10/1996	Lewis	395/154
5,572,643 A	11/1996	Judson	395/793
5,586,257 A	12/1996	Periman	463/42
5,586,260 A	12/1996	Hu	395/200.2
5,612,730 A	3/1997	Lewis	348/8
5,623,600 A	4/1997	Ji et al.	395/187.01
5,654,886 A	8/1997	Zereski, Jr. et al.	364/420
5,657,390 A	8/1997	Elgarni et al.	380/49
5,657,450 A	8/1997	Rao et al.	395/610

5,678,041 A	10/1997	Baker et al.	395/609
5,727,159 A	3/1998	Kikinis	395/200.76
5,842,216 A	11/1998	Anderson et al.	707/203
5,889,955 A	3/1999	Shinozaki et al.	395/200.54
5,918,013 A	* 6/1999	Mighdoli et al.	709/217
5,978,817 A	11/1999	Gianandrea et al.	707/501
5,996,022 A	* 11/1999	Krueger et al.	709/247
6,018,619 A	1/2000	Allard et al.	395/200.54
6,226,412 B1	5/2001	Schwab	382/232

OTHER PUBLICATIONS

Administrator's Guide, Netscape Proxy Server Version 2.0, Netscape Communications Corporation, pp. 19-20, 1996.
 Chankhuthod, Anawat, et al., "A Hierarchical Internet Object Cache," 1996 USEWIX Technical Conference (6 pages).
 Farrow, Rik, "Securing the Web: fire walls, proxy servers, and data driven attacks," InfoWorld, Jun. 19, 1995, vol. 7, No. 25, pp. 103-104.

* cited by examiner

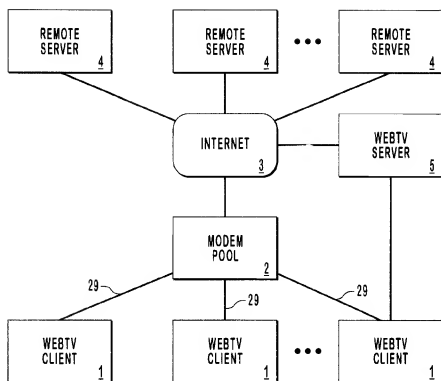


FIG. 1

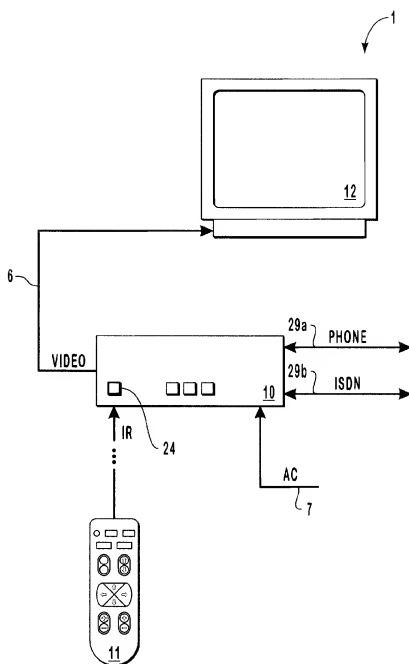


FIG. 2

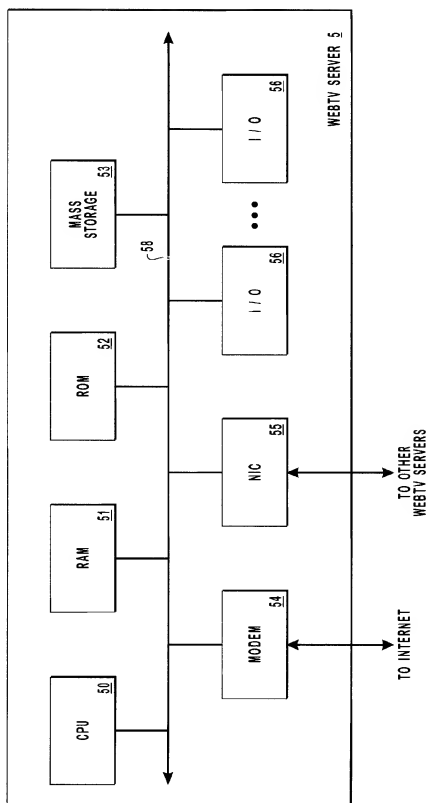


FIG. 3

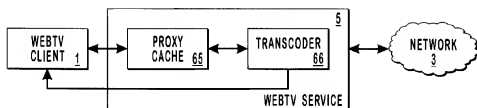


FIG. 4A

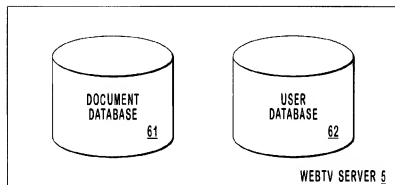


FIG. 4B

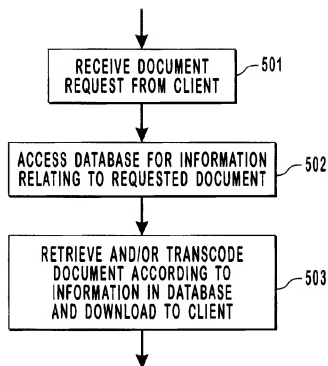


FIG. 5

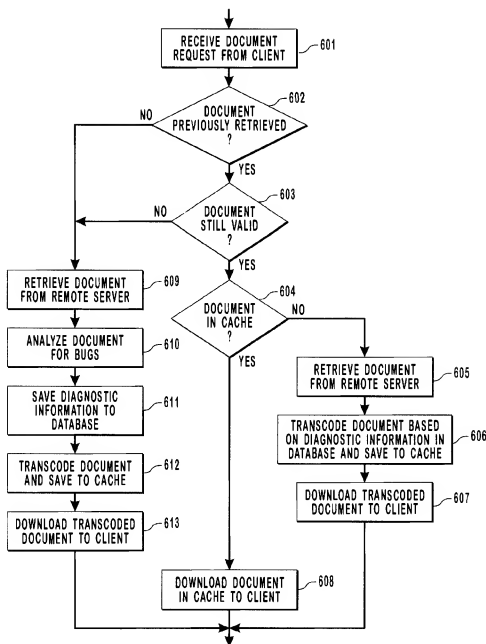


FIG. 6

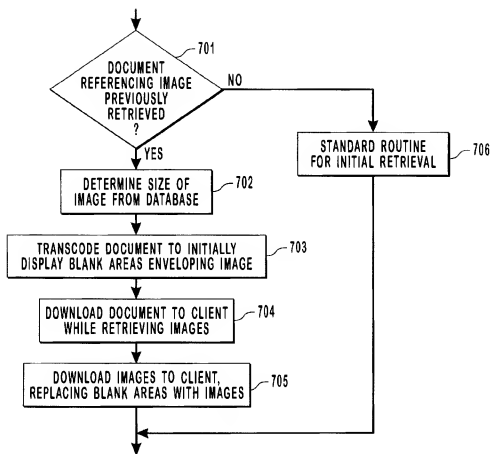


FIG. 7

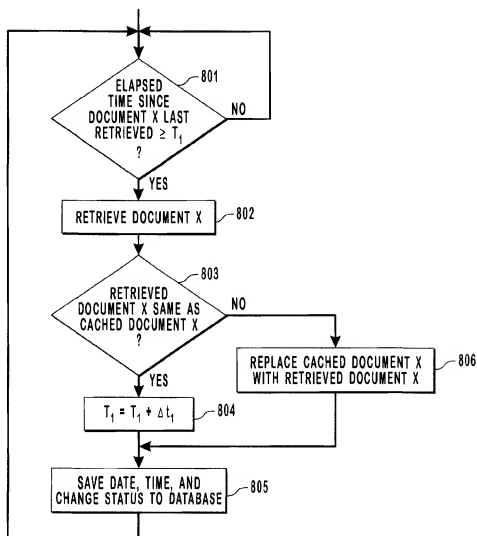


FIG. 8

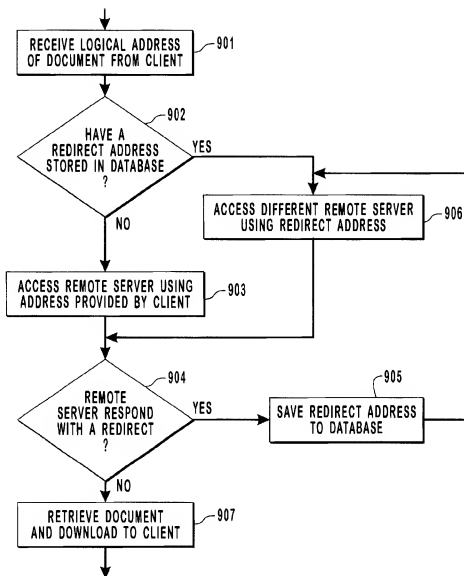


FIG. 9

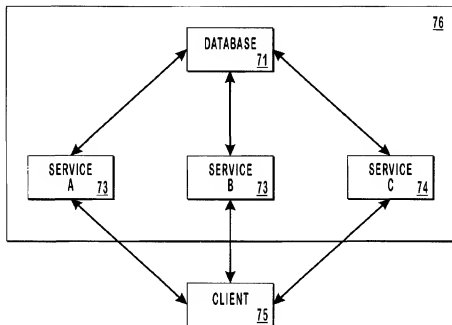


FIG. 10
(PRIOR ART)

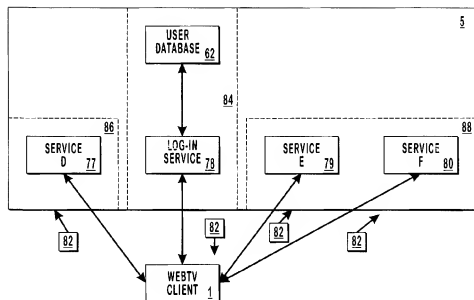


FIG. 11

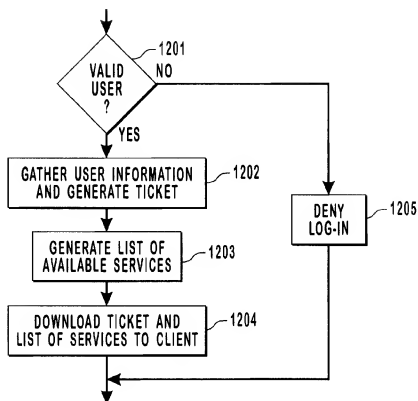


FIG. 12

1

METHOD OF TRANSCODING DOCUMENTS IN A NETWORK ENVIRONMENT USING A PROXY SERVER

RELATED APPLICATION

This is a continuation of U.S. patent application Ser. No. 09/343,067, entitled "Method of Transcoding Documents in a Network Environment Using a Proxy Server," filed Jun. 29, 1999, now abandoned which is a continuation of U.S. application Ser. No. 08/656,924 entitled "Method of Transcoding Documents in a Network Environment Using a Proxy Server," filed Jun. 3, 1996, now U.S. Pat. No. 5,918, 013 both of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. The Field of the Invention

The present invention pertains to the field of client-server computer networking. More particularly, the present invention relates to a method of transcoding documents in a network environment using a proxy server.

2. The Prior State of the Art

The number of people using personal computers has increased substantially in recent years, and along with this increase has come an explosion in the use of the Internet. One particular aspect of the Internet which has gained widespread use is the World-Wide Web ("the Web"). The Web is a collection of formatted hypertext pages located on numerous computers around the world that are logically connected by the Internet. Advances in network technology and software providing user interfaces to the Web ("Web browsers") have made the Web accessible to a large segment of the population. However, despite the growth in the development and use of the Web, many people are still unable to take advantage of this important resource.

Access to the Web has been limited thus far mostly to people who have access to a personal computer. However, many people cannot afford the cost of even a relatively inexpensive personal computer, while others are either unable or unwilling to learn the basic computer skills that are required to access the Web. Furthermore, Web browsers in the prior art generally do not provide the degree of user-friendliness desired by some people, and many computer novices do not have the patience to learn how to use the software. Therefore, it would be desirable to provide an inexpensive means by which a person can access the Web without the use of a personal computer. In particular, it would be desirable for a person to be able to access the Web pages using an ordinary television set and a remote control, so that the person feels more as if he or she is simply changing television channels, rather than utilizing a complex computer network.

Prior art Web technology also has other significant limitations which can make a person's experience unpleasant when browsing the Web. Web documents are commonly written in HTML (Hypertext Mark-up Language). HTML documents sometimes contain bugs (errors) or have features that are not recognized by certain Web browsers. These bugs or quirks in a document can cause a Web browser to fail. Thus, what is needed is a means for reducing the frequency with which client systems fail due to bugs or quirks in HTML documents.

Another problem associated with browsing the Web is latency. People commonly experience long, frustrating delays when browsing the Web. It is not unusual for a person to have to wait minutes after selecting a hypertext link for a

2

Web page to be completely downloaded to his computer and displayed on his computer screen. There are many possible causes for latency, such as heavy communications traffic on the Internet and slow response of remote servers. Latency can also be caused by Web pages including images. One reason for this effect is that, when an HTML document references an image, it takes time to retrieve the image itself after the referencing document has been retrieved. Another reason is that, in the prior art, if the referencing document does not specify the size of the image, the client system generally cannot display the Web page until the image itself has been retrieved. Numerous other sources of latency exist with respect to the Web. Therefore, what is needed is a means for reducing such latency, to eliminate some of the frustration which typically has been associated with browsing the Web.

Security is another concern associated with the Internet. Internet service providers (ISPs) generally maintain certain information about each customer in a database. This information may include information which a customer may not wish to become publicly known, such as social security numbers and credit card numbers. Maintaining the confidentiality of this information in a system that is connected to an expensive publicly-accessible computer network like the Internet can be problematic. Further, the problem can be aggravated by the fact that an ISP often provides numerous different services, each of which has access to this database. Allowing access to the database by many different entities creates many opportunities for security breaches to occur. Therefore, what is needed is a way to improve the security of confidential customer information in a server system coupled to the Internet.

SUMMARY AND OBJECTS OF THE INVENTION

A method is described of providing a document to a client coupled to a server. The server functions as a proxy on behalf of the client for purposes of accessing a remote server. In the method, a document is retrieved from the remote server in response to a request from the client. The document includes data to be used by the client in generating a display. The proxying server alters (i.e., transcodes) the data in the document to form a transcoded document. The transcoded document is then transmitted to the client.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follows. The proxying server transcodes the data in the document in order to perform at least one of the following functions: (1) matching decompression requirements at the client; (2) converting the document into a format compatible for the client; (3) reducing latency experienced by the client; and (4) altering the document to fit into smaller memory space.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follow.

BRIEF DESCRIPTION OF THE DRAWINGS

In order that the manner in which the above-recited and other advantages and objects of the invention are obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered limiting of its scope, the invention will be

3

described and explained with additional specificity and detail through the use of the accompanying drawings in which:

FIG. 1 illustrates several clients connected to a proxying server in a network;

FIG. 2 illustrates a client according to the present invention;

FIG. 3 is a block diagram of a server according to the present invention;

FIG. 4A illustrates a server including a proxy cache and a transcoder;

FIG. 4B illustrates databases used in a server according to the present invention;

FIG. 5 is a flow diagram illustrating a routine for transcoding a document retrieved from a remote server using data stored in a persistent database;

FIG. 6 is a flow diagram illustrating a routine for transcoding an HTML document for purposes of eliminating bugs or undesirable features;

FIG. 7 is a flow diagram illustrating a routine for reducing latency when downloading a document referencing an image to a client;

FIG. 8 is a flow diagram illustrating a routine for updating documents stored in the proxy cache using data stored in a persistent database;

FIG. 9 is a flow diagram illustrating a routine used by a server for retrieving documents from another remote server;

FIG. 10 is a block diagram of a prior art server system showing a relationship between various services and a database;

FIG. 11 is a block diagram of a server system according to the present invention showing a relationship between various services and a user database; and

FIG. 12 is a flow diagram illustrating a routine used by a server for regulating access to various services provided by the server.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

A method and apparatus are described for providing proxying and transcoding of documents in a network. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be evident, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

The present invention includes various steps, which will be described below. The steps can be embodied in machine-executable instructions, which can be used to cause a general-purpose or special-purpose processor programmed with the instructions to perform the steps. Alternatively, the steps of the present invention might be performed by specific hardware components that contain hardwired logic for performing the steps, or by any combination of programmed computer components and custom hardware components.

I. System Overview

The present invention is included in a system, known as WebTV™, for providing a user with access to the Internet. A user of a WebTV™ client generally accesses a WebTV™ server via a direct-dial telephone (POTS, for "plain old

4

telephone service"), ISDN (Integrated Services Digital Network), or other similar connection, in order to browse the Web, send and receive electronic mail (e-mail), and use various other WebTV™ network services. The WebTV™ network services are provided by WebTV™ servers using software residing within the WebTV™ servers in conjunction with software residing within a WebTV™ client.

FIG. 1 illustrates a basic configuration of the WebTV™ network according to one embodiment. A number of WebTV™ clients 1 are coupled to a modem pool 2 via direct-dial, bi-directional data connections 29, which may be telephone (POTS, i.e., "plain old telephone service"), ISDN (Integrated Services Digital Network), or any other similar type of connection. The modem pool 2 is coupled typically through a router, such as that conventionally known in the art, to a number of remote servers 4 via a conventional network infrastructure 3, such as the Internet. The WebTV™ system also includes a WebTV™ server 5, which specifically supports the WebTV™ clients 1. The WebTV™ clients 1 each have a connection to the WebTV™ server 5 either directly or through the modem pool 2 and the Internet 3. Note that the modem pool 2 is a conventional modem pool, such as those found today throughout the world providing access to the Internet and private networks.

Note that in this description, in order to facilitate explanation the WebTV™ server 5 is generally discussed as if it were a single device, and functions provided by the WebTV™ services are generally discussed as being performed by such single device. However, the WebTV™ server 5 may actually comprise multiple physical and logical devices connected in a distributed architecture, and the various functions discussed below which are provided by the WebTV™ services may actually be distributed among multiple WebTV™ server devices.

II. Client System

FIG. 2 illustrates a WebTV™ client 1. The WebTV™ client 1 includes an electronics unit 10 (hereinafter referred to as "the WebTV™ box 10"), an ordinary television set 12, and a remote control 11. In an alternative embodiment of the present invention, the WebTV™ box 10 is built into the television set 12 as an integral unit. The WebTV™ box 10 includes hardware and software for providing the user with a graphical user interface, by which the user can access the WebTV™ network services, browse the Web, send e-mail, and otherwise access the Internet.

The WebTV™ client 1 uses the television set 12 as a display device. The WebTV™ box 10 is coupled to the television set 12 by a video link 6. The video link 6 is an RF (radio frequency), S-video, composite video, or other equivalent form of video link. In the preferred embodiment, the client 1 includes both a standard modem and an ISDN modem, such that the communication link 29 between the WebTV™ box 10 and the server 5 can be either a telephone (POTS) connection 29a or an ISDN connection 29b. The WebTV™ box 10 receives power through a power line 7.

Remote control 11 is operated by the user in order to control the WebTV™ client 1 in browsing the Web, sending e-mail, and performing other Internet-related functions. The WebTV™ box 10 receives commands from remote control 11 via an infrared (IR) communication link. In alternative embodiments, the link between the remote control 11 and the WebTV™ box 10 may be RF or any equivalent mode of transmission.

III. Server System

The WebTV™ server 5 generally includes one or more computer systems generally having the architecture illus-

5

trated in FIG. 3. It should be noted that the illustrated architecture is only exemplary, the present invention is not constrained to this particular architecture. The illustrated architecture includes a central processing unit (CPU) 50, random access memory (RAM) 51, read-only memory (ROM) 52, a mass storage device 53, a modem 54, a network interface card (NIC) 55, and various other input/output (I/O) devices 56. Mass storage device 53 includes a magnetic, optical, or other equivalent storage medium. I/O devices 56 may include any or all of devices such as a display monitor, keyboard, cursor control device, etc. Modem 54 is used to communicate data to and from remote servers 4 via the Internet.

As noted above, the WebTV™ server 5 may actually comprise multiple physical and logical devices connected in a distributed architecture. Accordingly, NIC 55 is used to provide data communication with other devices that are part of the WebTV™ services. Modem 54 may also be used to communicate with other devices that are part of the WebTV™ services and which are not located in close geographic proximity to the illustrated device.

According to the present invention, the WebTV™ server 5 acts as a proxy in providing the WebTV™ client 1 with access to the Web and other WebTV™ services. More specifically, WebTV™ server 5 functions as a "caching proxy." FIG. 4A illustrates the caching feature of the WebTV™ server 5. In FIG. 4A, the WebTV™ server 5 is functionally located between the WebTV™ client 1 and the Internet infrastructure 3. The WebTV™ server 5 includes a proxy cache 65 which is functionally coupled to the WebTV™ client 1. The proxy cache 65 is used for temporary storage of Web documents, images, and other information which is frequently used by either the WebTV™ client 1 or the WebTV™ server 5.

A document transcoder 66 is functionally coupled between the proxy cache 65 and the Internet infrastructure 3. The document transcoder 66 includes software which is used to automatically revise the code of Web documents retrieved from the remote servers 4, for purposes which are described below.

The WebTV™ service provides a document database 61 and a user database 62, as illustrated in FIG. 4B. The user database 62 contains information that is used to control certain features relating to access privileges and capabilities of the user of the client 1. This information is used to regulate initial access to the WebTV™ service, as well as to regulate access to the individual services provided by the WebTV™ system, as will be described below. The document database 61 is a persistent database which stores certain diagnostic and historical information about each document and image retrieved by the server 5, as is now described.

A. Document Database

The basic purpose of the document database 61 is that, after a document has once been retrieved by the server 5, the stored information can be used by the server 5 to speed up processing and downloading of that document in response to all future requests for that document. In addition, the transcoding functions and various other functions of the WebTV™ service are facilitated by making use of the information stored in the document database 61, as will be described below.

Referring now to FIG. 5, the server 5 initially receives a document request from a client 1 (step 501). The document request will generally result from the user of the client 1

6

activating a hypertext anchor (link) on a Web page. The act of activating a hypertext anchor may consist of clicking on underlined text in a displayed Web page using a mouse, for example. The document request will typically (but not always) include the URL (Uniform Resource Locator) or other address of the selected anchor. Upon receiving the document request, the server 5 optionally accesses the document database 61 to retrieve stored information relating to the requested document (step 502). It should be noted that the document database 61 is not necessarily accessed in every case. The information retrieved from the document database 61 is used by the server 5 for determining, among other things, how long a requested document has been cached and/or whether the document is still valid. The criteria for determining validity of the stored document are discussed below. The server 5 retrieves the document from the cache 65 if the stored document is valid; otherwise, the server 5 retrieves the document from the appropriate remote server 4 (step 503). The server 5 automatically transcodes the document as necessary based on the information stored in the document database 61 (step 503). The transcoding functions are discussed further below.

The document database 61 includes certain historical and diagnostic information for every Web page that is accessed at any time by a WebTV™ client 1. As is well known, a Web page may correspond to a document written in a language such as HTML (Hypertext Mark-Up Language), VRML (Virtual Reality Modelling Language), or another suitable language. Alternatively, a Web page may represent an image, or a document which references one or more images. According to the present invention, once a document or image is retrieved by the WebTV™ server 5 from a remote server 4 for the first time, detailed information on this document or image is stored permanently in the document database 61. More specifically, for every Web page that is retrieved from a remote server 4, any or all of the following data are stored in the document database 61:

- 1) information identifying bugs (errors) or quirks in the Web page, or undesirable effects caused when the Web page is displayed by a client 1;
- 2) relevant bug-finding algorithms;
- 3) the date and time the Web page was last retrieved;
- 4) the date and time the Web page was most recently altered by the author;
- 5) a checksum for determining whether the Web page has been altered;
- 6) the size of the Web page (in terms of memory);
- 7) the type of Web page (e.g., HTML document, image, etc.);
- 8) a list of hypertext anchors (links) in the Web page and corresponding URLs;
- 9) a list of the most popular anchors based on the number of "hits" (requests from a client 1);
- 10) a list of related Web pages which can be prefetched;
- 11) whether the Web page has been redirected to another remote server 4;
- 12) a redirect address (if appropriate);
- 13) whether the redirect (if any) is temporary or permanent, and if permanent, the duration of the redirect;
- 14) if the Web page is an image, the size of the image in terms of both physical dimensions and memory space;
- 15) the sizes of in-line images (images displayed in text) referenced by the document defining the Web page;

7

- 16) the size of the largest image referenced by the document;
- 17) information identifying any image maps in the Web page;
- 18) whether to resize any images corresponding to the Web page;
- 19) an indication of any forms or tables in the Web page;
- 20) any unknown protocols;
- 21) any links to "dead" Web pages (i.e., pages which are no longer active);
- 22) the latency and throughput of the remote server 4 on which the Web page is located;
- 23) the character set of the document;
- 24) the vendor of the remote server 4 on which the Web page is located;
- 25) the geographic location of the remote server 4 on which the Web page is located;
- 26) the number of other Web pages which reference the subject Web page;
- 27) the compression algorithm used by the image or document;
- 28) the compression algorithm chosen by the transcoder;
- 29) a value indicating the popularity of the Web page based on the number of hits by clients; and
- 30) a value indicating the popularity of other Web pages which reference the subject Web page.

B. Transcoding

As mentioned above, the WebTV™ services provide a transcoder 66, which is used to rewrite certain portions of the code in an HTML document for various purposes. These purposes include: (1) correcting bugs in documents; (2) correcting undesirable effects which occur when a document is displayed by the client 1; (3) improving the efficiency of transmission of documents from the server 5 to the client 1; (4) matching hardware decompression technology within the client 1; (5) resizing images to fit on the television set 12; (6) converting documents into other formats to provide compatibility; (7) reducing latency experienced by a client 1 when displaying a Web page with in-line images (images displayed in text); and, (8) altering documents to fit into smaller memory spaces.

There are three transcoding modes used by the transcoder 66: (1) streaming, (2) buffered, and (3) deferred. Streaming transcoding refers to the transcoding of documents on a line-by-line basis as they are retrieved from a remote server 4 and downloaded to the client 1 (i.e., transcoding "on the fly"). Some documents, however, must first be buffered in the WebTV™ server 5 before transcoding and downloading them to the client 1. A document may need to be buffered before transmitting it to the client 1 if the type of changes to be made can only be made after the entire document has been retrieved from the remote server 4. Because the process of retrieving and downloading a document to the client 1 increases latency and decreases throughput, it is not desirable to buffer all documents. Therefore, the transcoder 66 accesses and uses information in the document database 61 relating to the requested document to first determine whether a requested document must be buffered for purposes of transcoding, before the document is retrieved from the remote server 4.

In the deferred mode, transcoding is deferred until after a requested document has been downloaded to a client 1. The deferred mode therefore reduces latency experienced by the

8

client 1 in receiving the document. Transcoding may be performed immediately after downloading or any time thereafter. For example, it may be convenient to perform transcoding during periods of low usage of WebTV™ services, such as at night. This mode is useful for certain types of transcoding which are not mandatory.

1. Transcoding for Bugs and Quirks

One characteristic of some prior art Web browsers is that they may experience failures ("crashes") because of bugs or unexpected features ("quirks") that are present in a Web document. Alternatively, quirks in a document may cause an undesirable result, even though the client does not crash. Therefore, the transcoding feature of the present invention provides a means for correcting certain bugs and quirks in a Web document. To be corrected by the transcoder 66, bugs and quirks must be identifiable by software running on the server 5. Consequently, the transcoder 66 will generally only correct conditions which have been previously discovered, such as those discovered during testing or reported by users. Once a bug or quirk is discovered, however, algorithms are added to the transcoder 66 to both detect the bug or quirk in the future in any Web document and to automatically correct it.

There are countless possibilities of bugs or quirks which might be encountered in a Web document. Therefore, no attempt will be made herein to provide an exhaustive list. Nonetheless, some examples may be useful at this point. Consider, for example, an HTML document that is downloaded from a remote server 4 and which contains a table having a width specified in the document as "0." This condition might cause a failure if the client were to attempt to display the document as written. This situation therefore, can be detected and corrected by the transcoder 66. Another example is a quirk in the document which causes quotations to be terminated with too many quotation marks. Once the quirk is first detected and an algorithm is written to recognize it, the transcoder 66 can automatically correct the quirk in any document.

If a given Web document has previously been retrieved by the server 5, there will be information regarding that document available in the document database 61 as described above. The information regarding this document will include whether or not the document included any bugs or quirks that required transcoding when the document was previously retrieved. The transcoder 66 utilizes this information to determine whether (1) the document is free of bugs and quirks, (2) the document has bugs or quirks which can be remedied by transcoding on the fly, or (3) the document has bugs or quirks which cannot be corrected on the fly (i.e., buffering is required).

FIG. 6 illustrates a routine for transcoding a Web document for purposes of eliminating bugs and quirks. Initially, the server 5 receives a document request from the client 1 (step 601). Next, the document database 61 is accessed to determine whether or not the requested document has been previously retrieved (step 602). If the document has not been previously retrieved, then the server 5 retrieves the document from the remote server 4 (step 603). Next, the retrieved document is analyzed for the presence of bugs or unusual conditions (step 610). Various diagnostic information is then stored in the document database 61 as a result of the analysis to note any bugs or quirks that were found (step 611). If any bugs or quirks were found which can be corrected by the transcoder 66, the document is then transcoded and saved to the proxy cache 65 (step 612). The transcoded document is

9

then downloaded to the client 1 (step 613). It should be noted that transcoding can be deferred until after the document has been downloaded, as described above; hence, the sequence of FIG. 6 is illustrative only.

If (in step 602) the requested document had been previously retrieved, then it is determined whether the requested document is still valid (step 603) and whether the document is present in the proxy cache 65 (step 604). If the document is no longer valid, then the document is retrieved from the remote server 4, analyzed for bugs and quirks, transcoded as required, and then downloaded to the client 1 as described above (steps 609-613). Methods for determining validity of a document are discussed below. If the document is still valid (step 603) and the document is present in the cache 65, the document is downloaded to the client 1 in its current form (as it is stored in the cache), since it has already been transcoded (step 608).

The document, however, may be valid but not present in the cache. This may be the case, for example, if the document has not been requested recently and the cache 65 has become too full to retain the requested document. In that case, the document is retrieved again from the remote server 4 (step 605) and then transcoded on the basis of the previously-acquired diagnostic information stored within the database 61 for that document. The document is then saved to the cache 65 (step 606). Note that because the document is still valid, it is assumed that the diagnostic information stored in the document database 61 for that document is still valid and that the transcoding can be performed on the basis of that information. Accordingly, once the document is transcoded, the transcoded document is downloaded to the client 1 (step 607). Again, note that transcoding can be deferred until after the document has been downloaded in some cases.

The validity of the requested document can be determined based on various different criteria. For example, some HTML documents specify a date on which the document was created, a length of time for which the document will be valid, or both. The validity determination can be based upon such information. For example, a document which specifies only the date of creation can be automatically deemed invalid after a predetermined period of time has passed.

Alternatively, validity can be based upon the popularity of the requested document. "Popularity" can be quantified based upon the number of hits for that document, which is tracked in the document database 61. For example, it might be prudent to simply assign a relatively short period of validity to a document which is very popular and a longer period of validity to a document which is less popular.

Another alternative basis for the validity of a document is the observed rate of change of the document. Again, data in the persistent document database 61 can be used. That is, because the document database 61 stores the date and time on which the document was last observed to change, the server 5 can approximate how often the document actually changes. A document or image which is observed to change frequently (e.g., a weather map or a news page) can be assigned a relatively short period of validity. It will be recognized that numerous other ways of determining validity are possible.

2. Transcoding to Reduce Latency

Another purpose for transcoding is to allow documents requested by a client 1 to be displayed by the client 1 more rapidly. Many HTML documents contain references to "in-line" images, or images that will be displayed in text in a

10

Web page. The normal process used in the prior art to display a Web page having in-line images is that the HTML document referencing the image is first downloaded to the client, followed by the client's requesting the referenced image. The referenced image is then retrieved from the remote server on which it is located and downloaded to the client. One problem associated with the prior art, however, is that the speed with which a complete Web page can be displayed to the user is often limited by the time it takes to retrieve in-line images. One reason for this is that it simply takes time to retrieve the image itself after the referencing document has been retrieved. Another reason is that, in the prior art, if the referencing document does not specify the size of the image, the Web page generally cannot be displayed until the image itself has been retrieved. The present invention overcomes these limitations.

According to the present invention, information stored in the document database 61 regarding the in-line images is used to transcode the referencing document in order to reduce latency in displaying the Web page. Once any document which references an in-line image is initially retrieved by the server 5, the fact that the document references an in-line image is stored in the document database 61. In addition, the size of the image is determined, either from the document (if specified) or from the image itself, and then stored in the document database 61. Consequently, for documents which do not specify the size of their in-line images, the size information stored in the database 61 is then used the next time the document is requested in order to reduce latency in downloading and displaying the Web page.

Refer now to FIG. 7, which illustrates a routine for reducing latency when downloading a document referencing an image to a client 1. Assume that a client 1 sends a request to the server 5 for an HTML document containing a reference to an in-line image. Assume further that the size of the image is not specified in the document itself. Initially, the server 5 determines whether that document has been previously retrieved (step 701). If not, the standard initial retrieval and transcoding procedure is followed (step 706), as described in connection with FIG. 6. If, however, the document has been previously retrieved, then the transcoder 66 accesses the size information stored in the document database 61 for the in-line image (step 702). Based on this size information, the HTML document is transcoded such that, when the Web page is initially displayed by the client 1, the area in which the image belongs is replaced by a blank region enveloping the shape of the image (step 703). Thus, any in-line image referenced by a document is displayed initially as a blank region. Consequently, the client 1 can immediately display the Web page corresponding to the HTML document even before the referenced image has been retrieved or downloaded (i.e., even before the size of the image is known to the client 1).

As the transcoded HTML document is downloaded to the client, the image is retrieved from the appropriate remote server 4 (step 704). Once the image is retrieved from the remote server 4 and downloaded to the client 1, the client 1 replaces the blank area in the Web page with the actual image (step 705).

3. Transcoding to Display Web Pages on a Television

As noted above, the client 1 utilizes an ordinary television set 12 as a display device. However, images in Web pages are generally formatted for display on a computer monitor, not a television set. Consequently, the transcoding function

11

of the present invention is used to resize images for display on the television set 12. This includes resealing images as necessary to avoid truncation when displayed on the television set 12.

It should be noted that prior art Web browsers which operate on computer monitors typically use resizable windows. Hence, the size of the visible region varies from client to client. However, because the web browser used by the WebTV™ client 1 is specifically designed for display on a television set, the present invention allows documents and images to be formatted when they are cached.

4. Transcoding for Transmission Efficiency

Documents retrieved by the server 5 are also transcoded to improve transmission efficiency. In particular, documents can be transcoded in order to reduce high frequency components in order to reduce interface flicker when they are displayed on a television set. Various methods for coding software or hardware to reduce perceptual interface flicker are described in co-pending U.S. patent application Ser. No. 08/656,923, filed on Jun. 3, 1996.

Documents can also be transcoded in order to lower the resolution of the displayed Web page. Reducing the resolution is desirable, because images formatted for computer systems will generally have a higher resolution than the NTSC (National Television Standards Committee) video format used by conventional television sets. Since the NTSC video does not have the bandwidth to reproduce the resolution of computer-formatted images, the bandwidth consumed in transmitting images to the client 1 at such a high resolution would be wasted.

5. Other Uses for Transcoding

Transcoding is also used by the present invention to recode a document using new formats into older, compatible formats. Images are often displayed in the JPEG (Joint Picture Experts Group) format or the GIF image format. JPEG often consumes less bandwidth than GIF, however. Consequently, images which are retrieved in GIF format are sometimes transcoded into JPEG format. Methods for generally converting images between GIF and JPEG formats are well known.

Other uses for transcoding include transcoding audio files. For example, audio may be transcoded into different formats in order to achieve a desired balance between memory usage, sound quality, and data transfer rate. In addition, audio may be transcoded from a file format (e.g., an "AU" file) to a streaming format (e.g., MPEG 1 audio). Yet another use of audio transcoding is the transcoding of MIDI (Musical Instrument Digital Interface) data to streaming variants of MIDI.

Additionally, documents or images requiring a large amount of memory (e.g., long lists) can be transcoded in order to consume less memory space in the client 1. This may involve, for example, separating a large document or image into multiple sections. For example, the server 5 can insert tags at appropriate locations in the original document so that the document appears to the client 1 as multiple Web pages. Hence, while viewing a given page representing a portion of the original document, the user can view the next page (i.e., the next portion of the original document) by activating a button on the screen as if it were an ordinary hypertext anchor.

C. Proxying

As noted above, the server 5 functions as a proxy on behalf of the client 1 for purposes of accessing the Web. The

12

document database 61 is used in various ways to facilitate this proxy role, as will now be described.

1. Updating Cached Documents

It is desirable to store frequently-requested HTML documents and images in the proxy cache 65 to further reduce latency in providing Web pages to the client 1. However, because some documents and images change over time, documents in the cache 65 will not be valid indefinitely, as mentioned above. A weather map or a news-related Web page, for example, are likely to be updated quite frequently. Consequently, it is desirable for the server 5 to have the ability to estimate the frequency with which documents change, in order to determine how long a document can safely remain within the proxy cache 65 without being updated.

The persistent database 65 is used to store the date and time of the last several fetches of each document and image retrieved from a remote server 4, along with an indication of any changes that were detected, if any. A document or image which has been stored in the cache 65 is then retrieved on a periodic basis to determine if it has been changed. Change status information indicating whether the document has changed since the previous fetch is then stored in the document database 61. If no changes are detected, then the time interval between fetches of this document is increased. If the document has changed, the time interval is maintained or decreased. As a result, items in the cache 65 which change frequently will be automatically updated at frequent intervals, whereas documents which do not change often will be replaced in the cache less frequently.

FIG. 8 illustrates a routine for updating documents stored in the proxy cache 65 using data stored in the document database 61. Assume a document X has been stored in the proxy cache 65. Document X remains in the cache 65 until a predetermined update period T_1 expires (step 801). Upon the expiration of the update period T_1 , the document X is again retrieved from the appropriate remote server 4 (step 802). The newly-retrieved document X is then compared to the cached version of document X (step 803). If the document has changed, then the cached version of document X is replaced with the newly-retrieved version of document X (step 806). If not, then the update period T_1 is increased according to a predetermined time increment ΔT_1 (step 804). In any case, the date and time and the change status of document X is saved to the document database 61 (step 805).

2. Document and Image Prefetching

The document database 61 is also used by the server 5 to store prefetching information relating to documents and images. In particular, the database stores, for each document that has been retrieved, a list of images referenced by the document, if any, and their locations. Consequently, the next time a document is requested by a client 1, the images can be immediately retrieved by the server 5 (from the cache 65, if available, or from the remote server 4), even before the client 1 requests them. This procedure improves the speed with which requested Web pages are downloaded to the client.

The document database 61 is also used to facilitate a process referred to as "server-advised client prefetching." Server-advised client prefetching allows the server 5 to inform the client 1 of documents or images which are popular to allow the client 1 to perform the prefetching. In particular, for any given document, a list is maintained in the

13

server 5 of the most popular hypertext anchors in that document (i.e., those which have previously received a large number of hits). When that document is requested by the client 1, the server 5 provides the client 1 with an indication of these popular links.

3. Redirects

Web pages are sometimes forwarded from the remote server on which they are initially placed to a different location. Under the HTTP (Hypertext Transport Protocol), such forwarding is sometimes referred to as a "redirect." When an HTML document is initially stored on one remote server and then later transferred to another remote server, the first remote server will provide, in response to a request for that document, an indication that the document has been transferred to a new remote server. This indication generally includes a forwarding address ("redirect address"), which is generally a URL.

In the prior art, when a computer requesting a Web page receives a redirect, it must then submit a new request to the redirect address. Having to submit a second request and wait for a second response consumes time and increases overall latency. Consequently, the present invention uses the document database 61 to store any redirect address for each document or image. Any time a redirected document is requested, the server 5 automatically accesses the redirect address to retrieve the document. The document or image is provided to the client 1 based on only a single request from the client 1. The change in location of the redirected document or image remains completely transparent to the client 1.

FIG. 9 illustrates a routine performed by the server 5 in accessing documents which may have been forwarded to a new remote server. Initially, the server 5 receives a request for a document, which generally includes an address (step 901). The server 5 then accesses the document database 61 to determine whether there is a redirect address for the requested document (step 902). If there is no redirect address, then the server 5 accesses a remote server 4 based on the address provided in the document request from the client 1 (step 903). Assuming that the remote server 4 does not respond to the server 5 with a redirect (step 904), the document is retrieved and downloaded to the client 1 by the server 5 (step 907). If, however, a redirect address was stored in the document database 61 (step 902), then the server 5 accesses the requested document according to the redirect address (step 906). Or, if the remote server 4 responded with a redirect (step 904), then the server 5 saves the redirect address to the document database 61 (step 905) and accesses the requested document according to the redirect address (step 906).

4. Other Proxy Functions

The document database 61 also stores information relating to the performance of each remote server 4 from which a document is retrieved. This information includes the latency and throughput of the remote server 4. Such information can be valuable in instances where a remote server 4 has a history of responding slowly. For example, when the document is requested, this knowledge can be used by the server 5 to provide a predefined signal to the client 1. The client 1 can, in response to the signal, indicate to the user that a delay is likely and give the user the option of canceling the request.

5. Backoff Mode

Although the server 5 generally operates in the proxy mode, it can also enter a "backoff mode" in which the server

14

5 does not act as a proxy, or the server 5 performs only certain aspects of the normal proxying functions. For example, if the proxy cache 65 is overloaded, then the server 5 can enter a backoff mode in which documents are not cached but are transcoded as required. Alternatively, during times when the server 5 is severely overloaded with network traffic, the server 5 may instruct the client 1 to bypass the server 5 and contact remote servers 4 directly for a specified time or until further notice. Or, the server 5 can enter a flexible backoff mode in which the client 1 will be instructed to contact a remote server 4 directly only for certain Web sites for a limited period of time.

D. Access to WebTV™ Services

The WebTV™ server 5 provides various services to the client 1, such as proxying and electronic mail ("e-mail"). In the prior art, certain difficulties are associated with allowing a client computer access to different services of an Internet service, as will now be explained with reference to FIG. 10.

FIG. 10 illustrates a client-server system according to one prior art embodiment. The server 76 provides various services A, B, and C. The server 76 includes a database 71 for storing information on the user's access privileges to services A, B, and C. The client 75 of the embodiment of FIG. 10 accesses any of services A, B, and C by contacting that service directly. The contacted service then accesses the database 71, which stores the access privileges of the client 75, to determine whether the client 75 should be allowed to access that service. Hence, each service provided by the server 76 requires direct access to the database 71. This architecture results in a large number of accesses being made to the database 71, which is undesirable. In addition, the fact that each service independently has access to the database 71 raises security concerns. Specifically, it can be difficult to isolate sensitive user information. The present invention overcomes such difficulties using a technique which is now described.

1. Tickets Containing Privileges And Capabilities

As shown in FIG. 11, the server 5 provides a number of services D, E, and F 77, 79, and 80, respectively, and a log-in service 78. The log-in service 78 is used specifically to control initial log-on procedures by a client 1. The log-in service 78 has exclusive access to the user database 62 (discussed above with respect to FIG. 4B). The log-in service 78 and the user database 62 are located within a first security zone 84. Service D is located within a second security zone 86, while services E and F are contained within a third security zone 88. Note that the specific arrangement of security zones 84, 86, and 88 with respect to services D, E, and F is illustrative only.

The user database 62 of the present invention stores various information pertaining to each authorized user of a client 1. This information includes account information, a list of the WebTV™ services that are available to the particular user, and certain user preferences. For example, a particular user may not wish his client 1 to be used to access Web pages having adult-oriented subject matter. Consequently, the user would request that his account be filtered to prevent access to such material. This request would then be stored as part of the user data in the user database 62.

With regard to user preferences, the hypertext links selected by a given user can be tracked, and those having the largest number can be stored in the user database 62. The list can then be provided to the client 1 for use in generating a

15

menu screen of the user's favorite Web sites, to allow the user to directly access those Web sites. The list can also be used by the server 5 to analyze the user's interests and to formulate and provide to the user a list of new Web sites which the user is likely to be interested in. The list might be composed by associated key words in Web pages selected by the user with other Web pages.

Referring again to FIG. 11, in response to a log-on request by a client 1, the log-in service 78 consults the user database 62 to determine if access to the server 5 by this particular client 1 is authorized. Assuming access is authorized, the log-in service 78 retrieves certain user information pertaining to this particular client 1 from the user database 62. The log-in service then generates a "ticket" 82, which is an information packet including the retrieved information. The ticket 82 is then provided to the client 1 which requested access.

The ticket 82 includes all information necessary to describe the access privileges of a particular user with respect to all services provided by the server 5. For example, the ticket may include the user name registered to the client 1, the e-mail address assigned to client 1, and any filtering requested by the user with respect to viewing Web sites. Each time the user requests access to one of the services D, E, or F, the client 1 submits a copy of the ticket 82 to that service. The requested service can then determine from the copy of the ticket 82 whether access to that service by that client 1 is authorized and, if so, any important information relating to such access.

None of the services provided by the server 5, other than the log-in service 78, has access to the user database 62. Hence, any security-sensitive information can be isolated within the user database 62 and the log-in service 78. Such isolation allows the individual services provided by the server 5 to be placed within separate "firewalls" (security regions), illustrated as security zones 84, 86, and 88. In addition, this technique greatly reduces the number of accesses required to the user database 62 compared to the prior art embodiment illustrated in FIG. 10.

2. Redundancy of Services and Load Balancing

The present invention also includes certain redundancies in the various services provided by the server 5. In particular, a given service (e.g., e-mail) can be provided by more than one physical or logical device. Each such device is considered a "provider" of that service. If a given provider is overloaded, or if the client 1 is unable to contact that provider, the client 1 can contact any of the other providers of that service. When the server 5 receives a log-in request from a client 1, in addition to generating the above-described ticket 82, the log-in service 78 dynamically generates a list of available WebTV™ services and provides this list to the client 1.

The server 5 can update the list of services used by any client 1 to reflect services becoming unavailable or services coming on-line. Also, the list of services provided to each client 1 can be updated by the server 5 based upon changes in the loading of the server 5, in order to optimize traffic on the server 5. In addition, a client's list of services can be updated by services other than the log-in service 78, such that one service can effectively introduce another service to the client 1. For example, the e-mail service may provide a client 1 with the name, port number and IP of its address book service. Thus, one service can effectively, and securely within the same chain of trust, introduce another service to the client 1.

16

This list of services includes the name of each service, a port number for the provider of each service, and an IP (Internet Protocol) for each service. Different providers of the same service are designated by the same name, but different port numbers and/or IPs. Note that in a standard URL, the protocol is normally specified at the beginning of the URL, such as "HTTP://www. . . ." under the HTTP protocol. However, according to the present invention, the normal protocol designation (i.e., "HTTP") in the URL is replaced with the name of the service, since the port number and IP for each service are known to the client 1. Hence, the client 1 can access any of the redundant providers of a given service using the same URL. This procedure effectively adds a level of indirection to all accesses made to any WebTV™ service and automatically adds redundancy to the proxy service. It should also be noted that separate service names can also refer to the same service.

Assume, for example, that the e-mail service provided by the WebTV™ system is designated by the service name "WTV-mailto." A client 1 can access any provider of this e-mail service using the same URL. The client 1 merely chooses the appropriate port number and IP number to distinguish between providers. If the client 1 is unable to connect to one e-mail provider, it can simply contact the next one in the list.

Thus, at log-in time, a client 1 is provided with both a ticket containing privileges and capabilities as well as a list of service providers, as illustrated in FIG. 12. Initially, the log-in service 78 determines whether the user of client 1 is a valid user (step 1201). If not, log-in is denied (step 1205). If the user is a valid user, then the log-in service 78 gathers user information from the user database 62 and generates a ticket 82 (step 1202). The log-in service 78 also generates the above-described list of services (step 1203). The ticket 82 and the list of services are then downloaded to the client 1 (step 1204).

3. Asynchronous Notification to Clients by Server

Another limitation associated with prior art Internet servers is the inability to provide asynchronous notification information to the client in the absence of a request from the client to do so. It would be desirable, for example, for a server to notify a client on its own initiative when a particular Web page has changed or that a particular service is inaccessible. The server 5 of the present invention provides such capability, and the client 1 is configured to receive and decode such notifications. For example, the client 1 can receive updates of its listing of service providers from the server 5 at various points in time, as already described. Similarly, if a particular service provider becomes unavailable, that fact will be automatically communicated to the client 1. As another example, if e-mail addressed to the user has been received by the server 5, then the server 5 will send a message to the client 1 indicating this fact. The client 1 will then notify the user that e-mail is waiting by a message displayed on the television set 12 or by an LED (light emitting diode) built into the housing of WebTV™ box 10.

Thus, a method and apparatus have been described for providing proxying and transcoding of documents in a network. The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

17

What is claimed and desired to be secured by United States Letters Patent is:

1. In a proxying server coupled to a client and to a remote server, the proxying server operating as a proxy on behalf of the client for accessing the remote server, a method of retrieving and transcoding a document requested by the client, the method comprising the steps of:

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document in order to perform at least one function selected from the following functions:

matching decompression requirements at the client; converting the document into a format compatible for the client;

reducing latency experienced by the client; and altering the document to fit into smaller memory space; and

transmitting the document to the client.

2. A method according to claim 1, wherein the request includes a URL (Uniform Resource Locator).

3. A method according to claim 1, wherein the step of transcoding is performed while the step of retrieving the document is performed.

4. A method according to claim 1, wherein the step of transcoding comprises the step of reading at least a portion of the document from a proxy cache located in the proxying server.

5. A method according to claim 1, wherein the step of transmitting the document to the client is performed prior to performing the step of transcoding, at the proxying server, the data in the document.

6. A method according to claim 1, wherein the step of transmitting the document to the client comprises the step of transmitting the transcoded document to the client, wherein the step of transmitting the document to the client is performed after the step of transcoding, at the proxying server, the data in the document.

7. A method according to claim 1, wherein the document includes a link to another document, the link including a retrieval address, and wherein the step of transcoding at the proxying server the data in the document comprises the step of updating the link

8. A method according to claim 7, wherein the other document is an image, and wherein the step of updating the link comprises the step of adding information to the document indicating the size of the image.

9. A method according to claim 8, wherein the other document is inaccessible to the proxying server, and wherein the step of updating the link comprises the step of removing the link.

10. A method according to claim 7, wherein the other document has been relocated from the retrieval address to a redirect address, and wherein the step of updating the link comprises the step of updating the link to correspond to the redirect address.

11. A method according to claim 1, wherein the client includes a television display, wherein the document references an image, and wherein the step of transcoding, at the proxying server, the data in the document is conducted in order to perform at least the function of converting the document into a compatible format for the client, and comprises the step of revising the data such that the image is sized for display on the television display.

18

12. A method according to claim 1, wherein the document references an image having a first image format, and wherein the step of transcoding, at the proxying server, the document is conducted in order to perform at least the function of converting the document into a compatible format for the client, and includes the step of converting the first image format to a second image format.

13. A method according to claim 1, further comprising the steps of:

identifying an image referenced by the document;

determining whether the image has been previously retrieved by the proxying server; and

if the image has been previously retrieved by the proxying server, accessing information stored in the proxying server indicating the size of the image;

wherein the step of transcoding, at the proxying server, the data in the document is conducted in order to perform at least the function of reducing latency experienced by the client, and comprises the step of using the information indicating the size of the image to revise the data of the document to allow the document to be partially displayed by the client before the image is received by the client.

14. A method according to claim 13, wherein the step of transcoding, at the proxying server, the data in the document comprises the following step:

if the image has been previously retrieved by the proxying server, transcoding at the proxying server the data in the document without the image but providing a space for the image to be later inserted; and

wherein the step of transmitting the document to the client comprises the following steps:

transmitting the document to the client without the image but providing the space for the image to be later inserted; and

after transmitting the document to the client without the image, transmitting the image to the client.

15. In a proxying server coupled to a client and to a remote server, the proxying server operating as a proxy on behalf of the client for accessing the remote server, a method of retrieving and transcoding a document requested by the client, the method comprising the steps of:

providing a persistent database at the proxying server, the persistent database including information relating to the document;

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document using the information included in the persistent database in order to perform at least one function selected from the following functions:

matching decompression requirements at the client; converting the document into a format compatible for the client;

reducing latency experienced by the client; and altering the document to fit into smaller memory space; and

transmitting the document to the client.

16. A method according to claim 15, further comprising the step of storing in the persistent database validity information corresponding to the document.

19

17. A method according to claim 16, wherein the step of retrieving the document comprises the following steps:

retrieving the document from the persistent database if the validity information corresponding to the document indicates that the document is valid; and

retrieving the document from the remote server if the validity information corresponding to the document indicates that the document is not valid.

18. A method according to claim 16, wherein the step of storing in the persistent database validity information corresponding to the document comprises the step of observing a rate of change of the document.

19. A method according to claim 18, wherein the step of observing a frequency of change of the document comprises the step of periodically retrieving the document, wherein successive retrievals of the document are separated in time by at least a time interval.

20. A method according to claim 19, wherein the step of periodically retrieving the document comprises the following steps:

retrieving the document at a first time;

storing the document retrieved at the first time in a memory;

retrieving the document at a second time, the second time being at least the time interval after the first time;

determining whether the document retrieved at the second time has changed compared to the document retrieved at the first time;

if the document retrieved at the second time is different than the document retrieved at the first time, replacing the document retrieved at the first time with the document retrieved at the second time in the memory; and

if the document retrieved at the second time is the same as the document retrieved at the first time, extending the time interval.

21. A method according to claim 15, further comprising the step of storing in the persistent database performance information relating to performance of the remote server when accessing the document.

22. A method according to claim 21, wherein the performance information comprises a latency value.

23. A method according to claim 15, further comprising the step of storing in the persistent database information for optimizing memory usage by the client.

24. A method according to claim 15, wherein the step of retrieving the document comprises the following steps:

determining whether a redirect address corresponding to the document is stored in the persistent database;

if the redirect address corresponding to the document is stored in the persistent database, accessing the remote server using the redirect address, the remote server corresponding to the redirect address; and

if the redirect address corresponding to the document is not stored in the persistent database, the method further comprises the following steps:

accessing a previous remote server at which the document previously resided;

obtaining the redirect address from the previous remote server;

storing the redirect address in the persistent database; and

20

accessing the remote server using the redirect address.

25. A computer program product for implementing, in a proxying server coupled to a client and to a remote server, a method of retrieving and transcoding a document requested by the client, the computer program product comprising a computer-readable medium carrying computer-executable instructions for causing the proxying server to perform acts included in the method, said acts comprising:

receiving, at the proxying server, a request for the document from the client, wherein the document resides at the remote server;

retrieving the document, the document including data for causing the client to generate a display of the document;

transcoding, at the proxying server, at least a portion of the document in order to perform at least one function selected from the following functions:

matching decompression requirements at the client;

converting the document into a format compatible for the client;

reducing latency experienced by the client; and
altering the document to fit into smaller memory space; and

transmitting the document to the client.

26. A computer program product according to claim 25, wherein the computer-executable instructions comprise a plurality of program code means for transcoding at least a portion of the document, including:

program code means for transcoding at least a portion of the document so as to match decompression requirements at the client;

program code means for transcoding at least a portion of the document so as to convert the document into a format compatible with the client;

program code means for transcoding at least a portion of the document so as to reduce latency experienced by the client; and

program code means for transcoding at least a portion of the document so as to alter the document to fit into smaller memory space.

27. A computer program product according to claim 26, wherein the act of transcoding comprises selecting at least one of said plurality of program code means for transcoding for use in the act of transcoding.

28. A computer program product according to claim 25, wherein the act of transcoding is performed while the act of retrieving the document is performed.

29. A computer program product according to claim 25, wherein the act of transcoding comprises reading at least a portion of the document from a proxy cache located in the proxying server.

30. A computer program product according to claim 25, wherein the act of transmitting the document to the client is performed prior to performing the act of transcoding.

31. A computer program product according to claim 25, wherein the act of transmitting the document to the client comprises the act of transmitting the transcoded document to the client, wherein the act of transmitting the document to the client is performed after the act of transcoding.

* * * * *



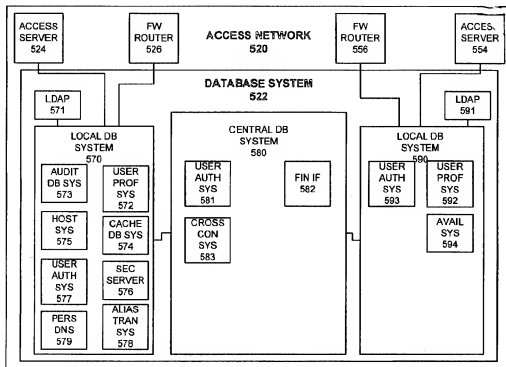
US006697806B1

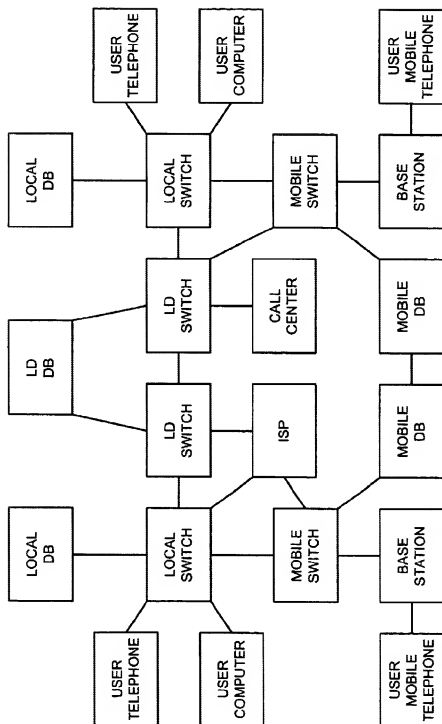
**(12) United States Patent
Cook****(10) Patent No.: US 6,697,806 B1
(45) Date of Patent: Feb. 24, 2004****(54) ACCESS NETWORK AUTHORIZATION****(75) Inventor: Fred S. Cook, Olathe, KS (US)****(73) Assignee: Sprint Communications Company,
L.P., Overland Park, KS (US)****(*) Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.**(21) Appl. No.: 09/574,978****(22) Filed: May 19, 2000****Related U.S. Application Data****(63)** Continuation of application No. 09/556,276, filed on Apr. 24, 2000.**(51) Int. Cl.⁷ G06F 17/30****(52) U.S. Cl. 707/10; 707/3; 707/9;
709/227; 709/203; 709/219****(58) Field of Search 707/1-3, 9-10,
707/4, 104.1; 709/223-227, 219, 203****(56) References Cited****U.S. PATENT DOCUMENTS**5,884,312 A * 3/1999 Dustan et al. 707/10
6,151,601 A * 11/2000 Papierniak et al. 707/1

* cited by examiner

*Primary Examiner—Jean R. Homere
Assistant Examiner—Mohammad Ali***(57) ABSTRACT**

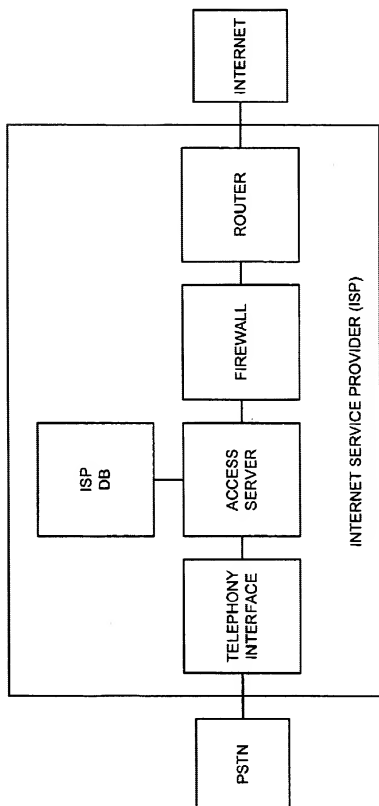
An access communication system provides access between a user system and a plurality of communication networks. The plurality of communication networks provide services to a user in the user system. An access communication system includes a local database system and an access server that is connected to the user system and the plurality of communication networks. The local database system receives a user logon. The local database system then processes the user logon to determine if the user is allowed access to the access communication system based on a local database system. The local database system then provides access to the access communication system to the user in response to the determination that the user is allowed access based on the local database system. The local database system then generates an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system. The local database system receives and processes an authorization response indicating whether the user is allowed to use the access system from the second database system. The local database system then provides access to the access communication system to the user in response to the authorization response that allows the user to use the access communication system.

24 Claims, 63 Drawing Sheets



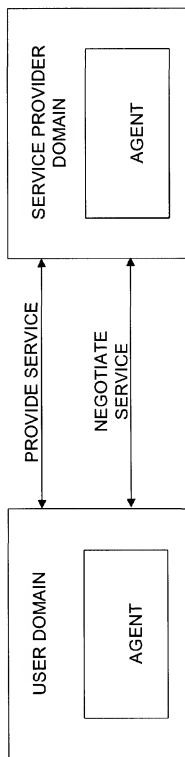
PRIOR ART

FIG. 1



PRIOR ART

FIG. 3



PRIOR ART

FIG. 4

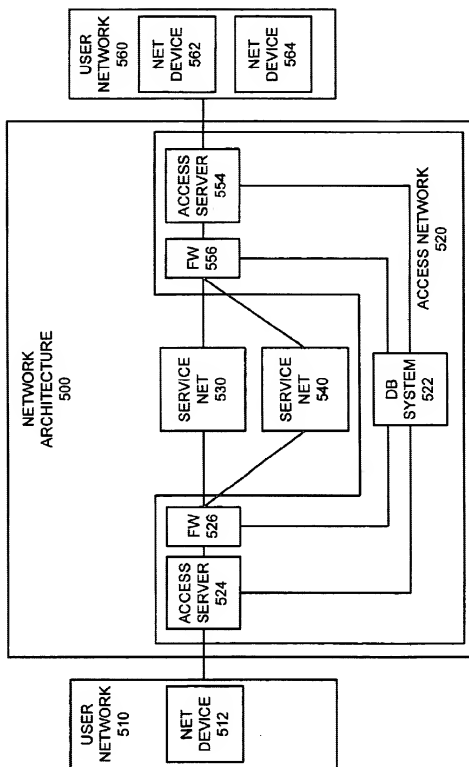


FIG. 5

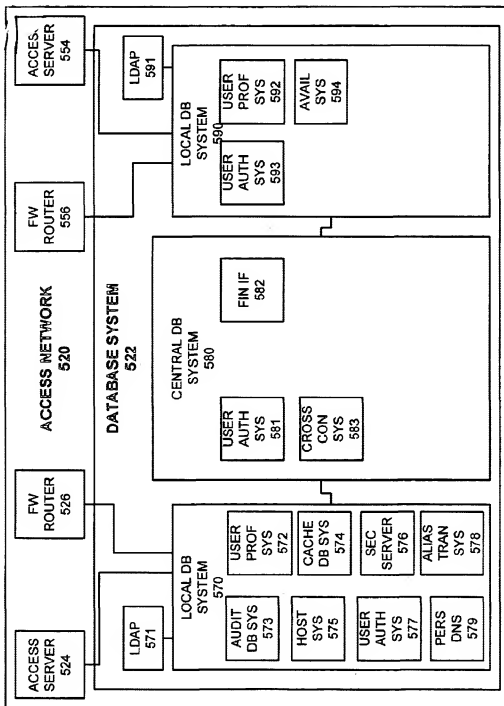


FIG. 6

USER ID	PASSWORD	NAME	ACCOUNT NUMBER	SERVICES	ADDRESS	BILLING CODE	CLASS	GROUP	SHELL	MACROS

FIG. 7

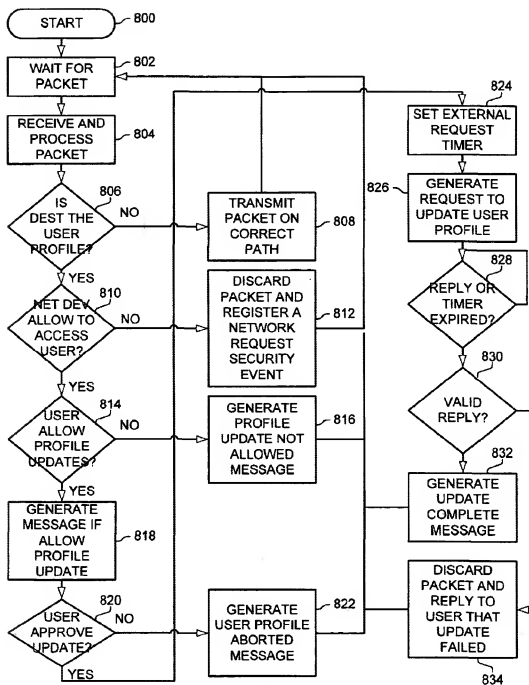


FIG. 8

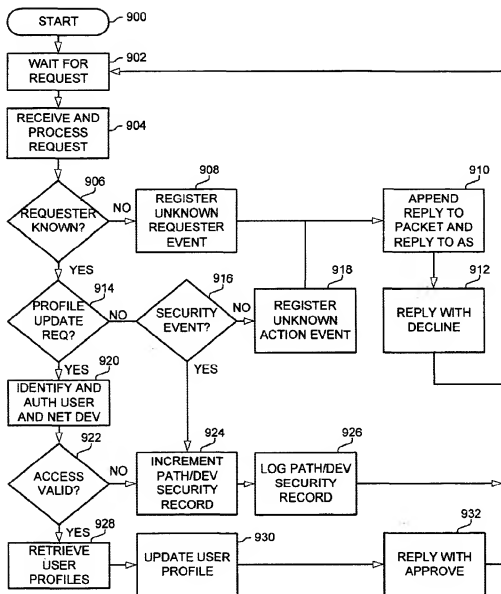


FIG. 9

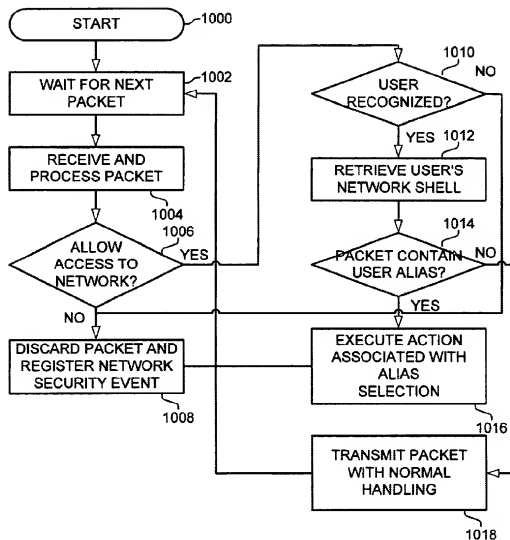


FIG. 10

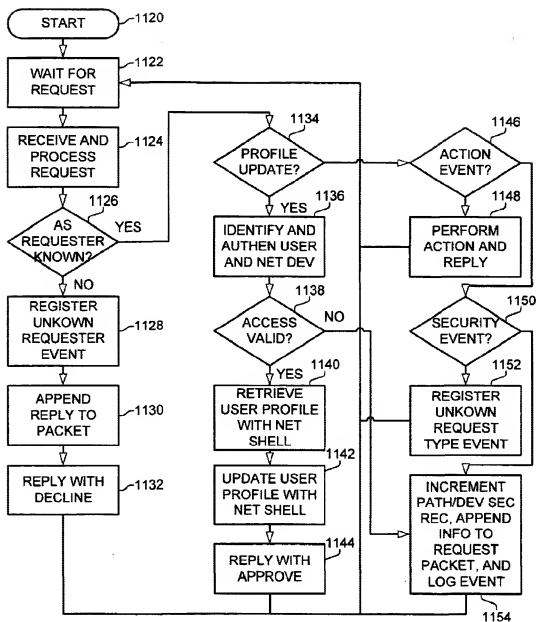


FIG. 11

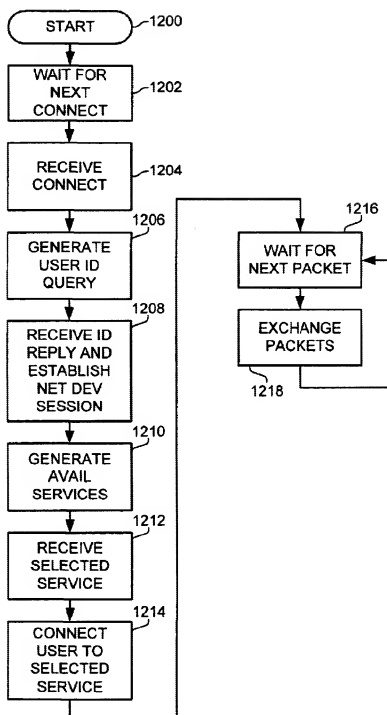


FIG. 12

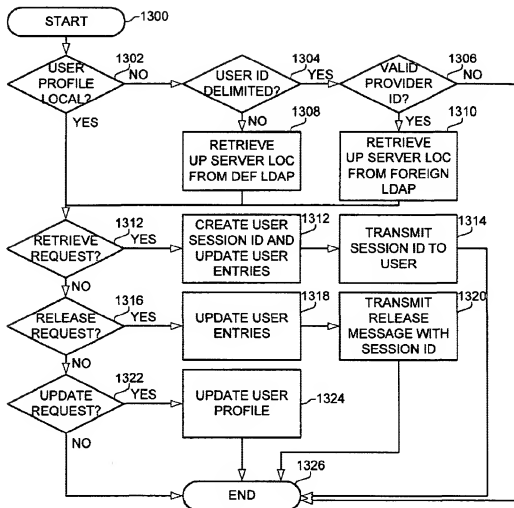


FIG. 13

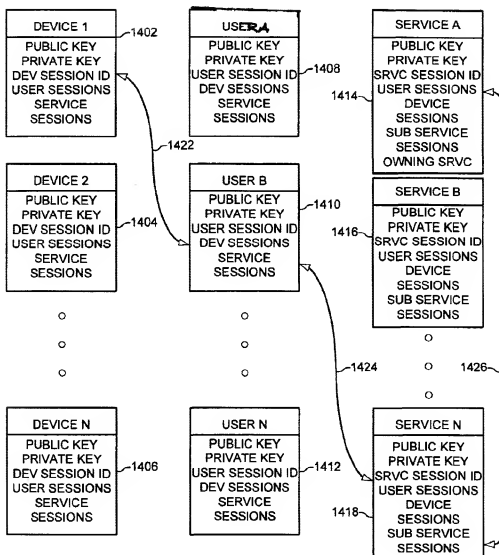
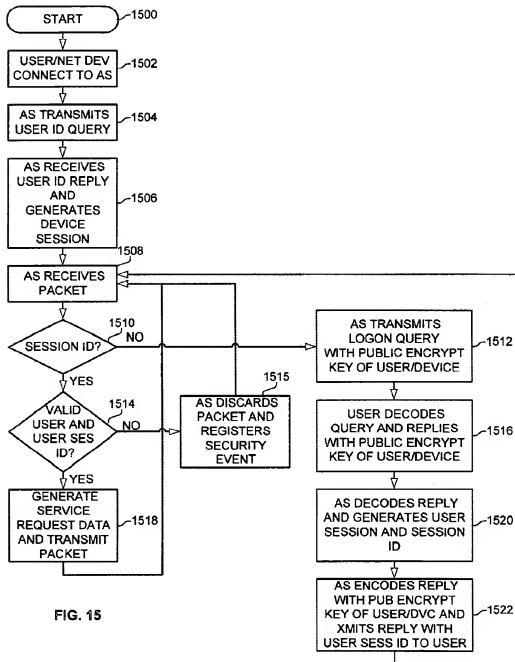


FIG. 14



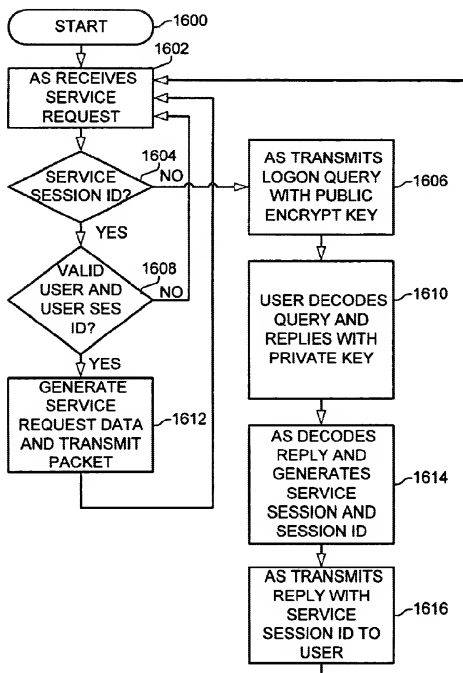


FIG. 16

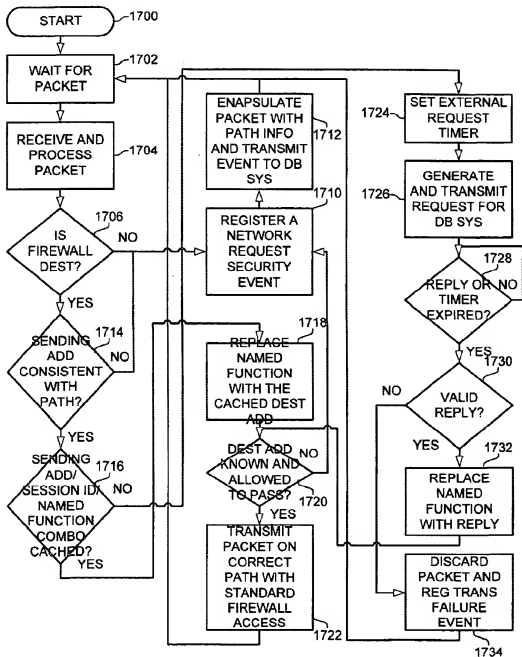


FIG. 17

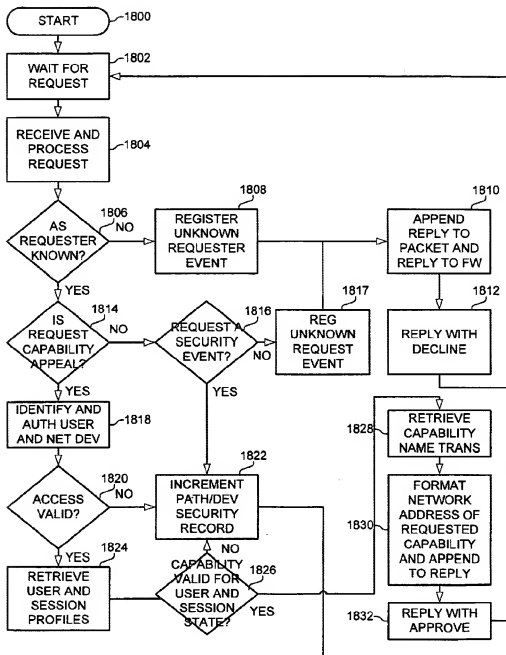


FIG. 18

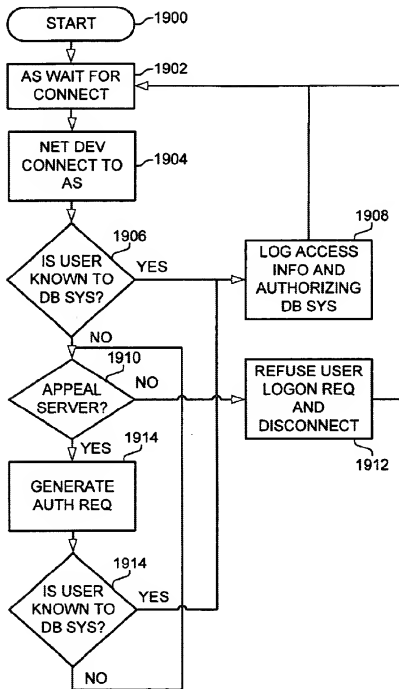


FIG. 19

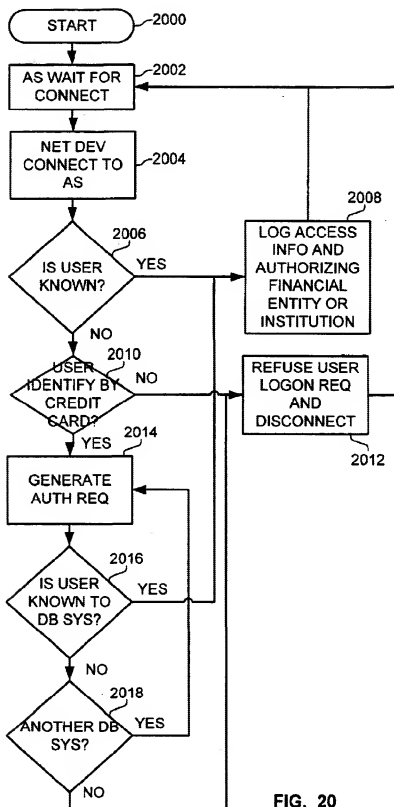


FIG. 20

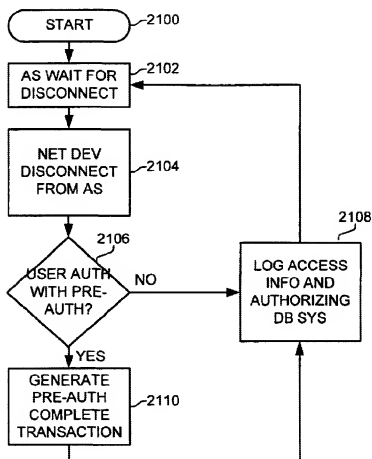


FIG. 21

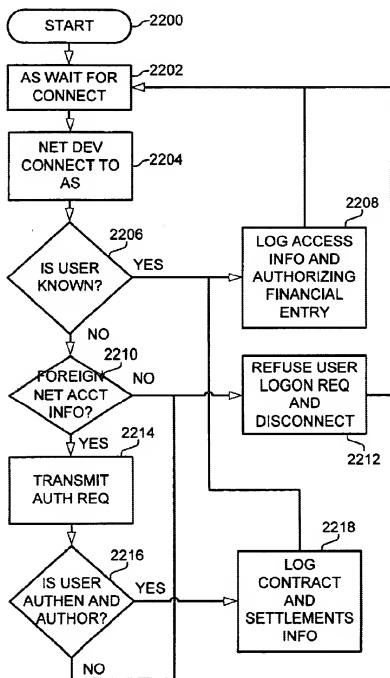
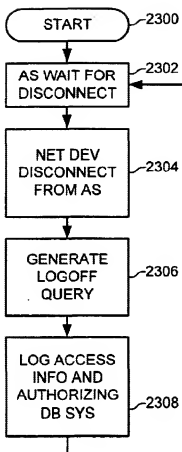


FIG. 22

**FIG. 23**

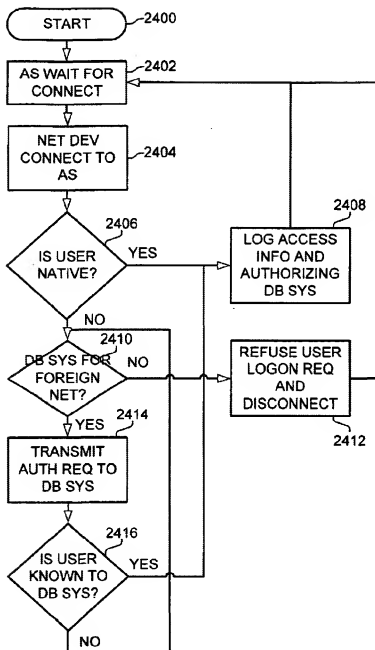


FIG. 24

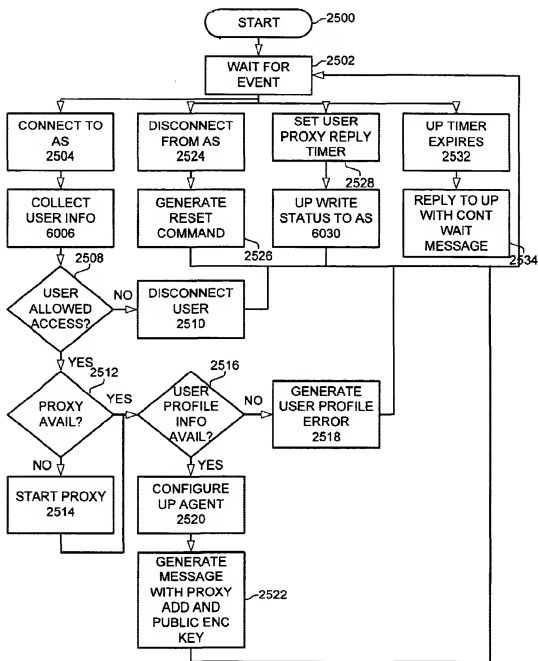


FIG. 25

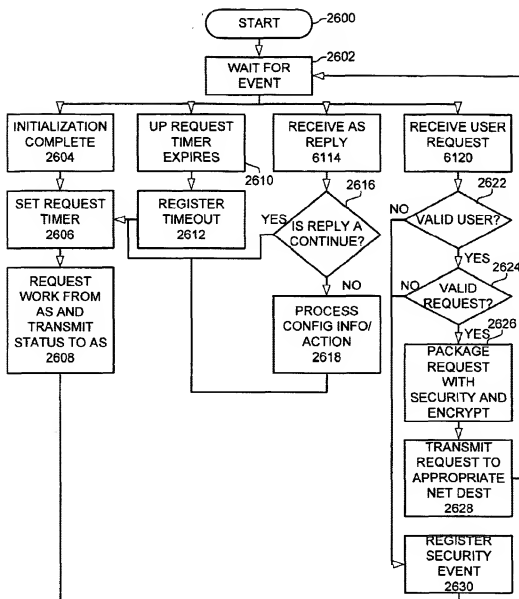


FIG. 26

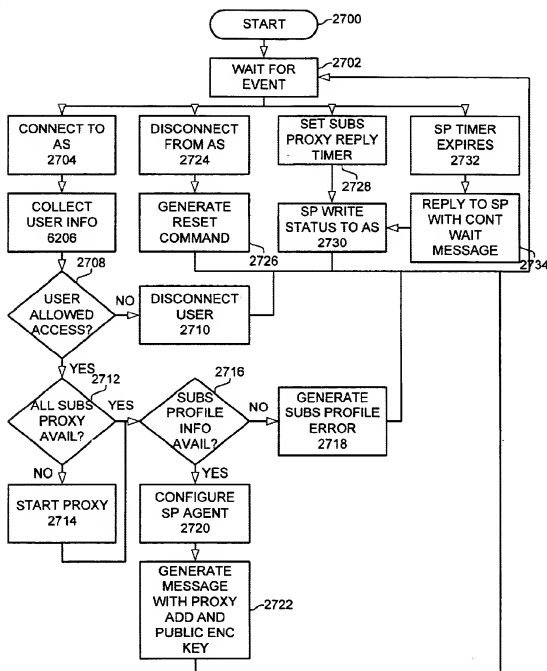


FIG. 27

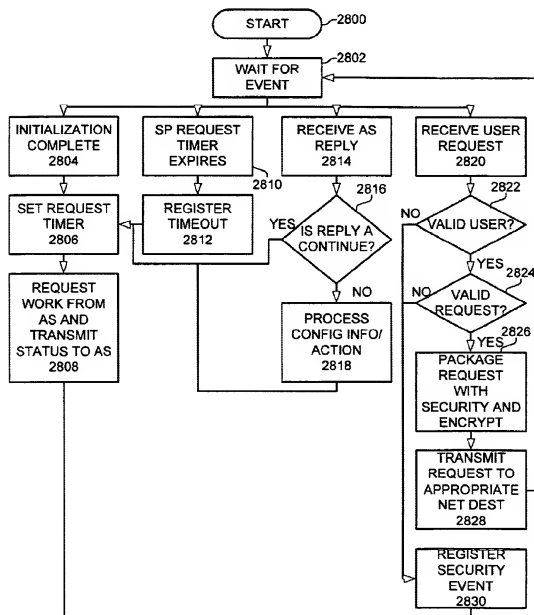


FIG. 28

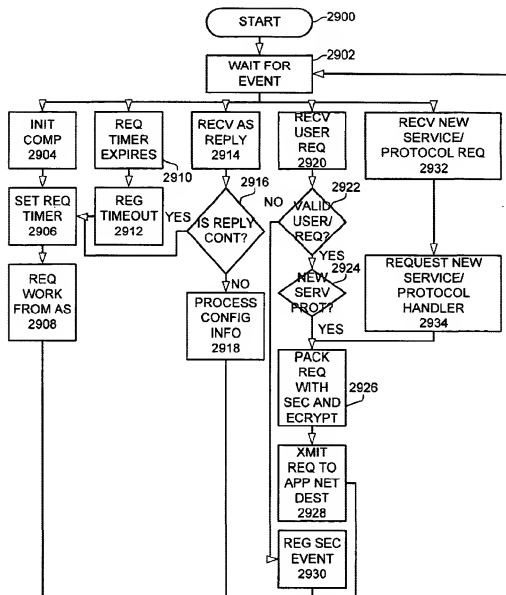


FIG. 29

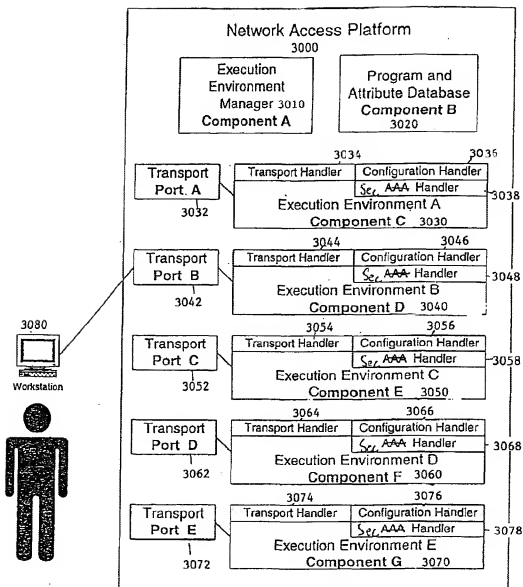


FIGURE 30

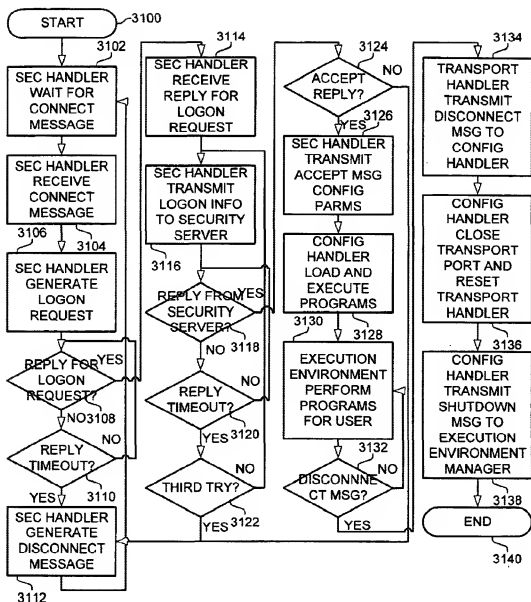


FIG. 31

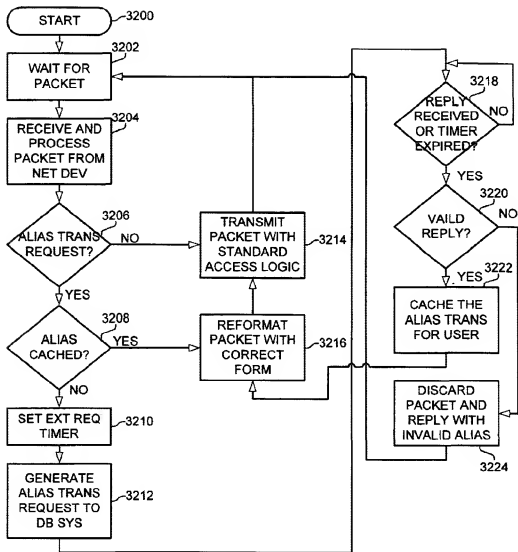


FIG. 32

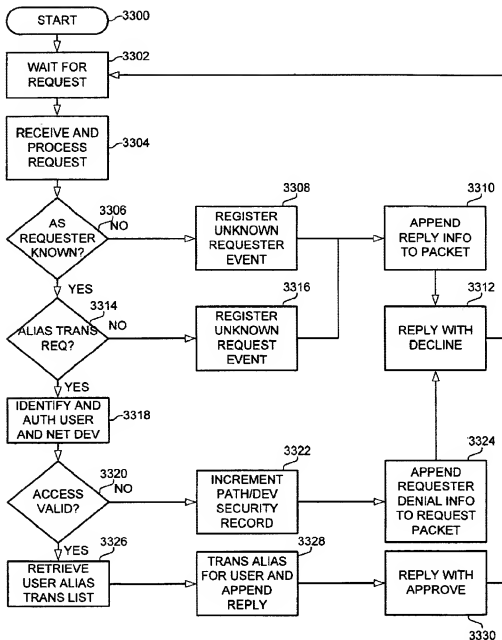


FIG. 33

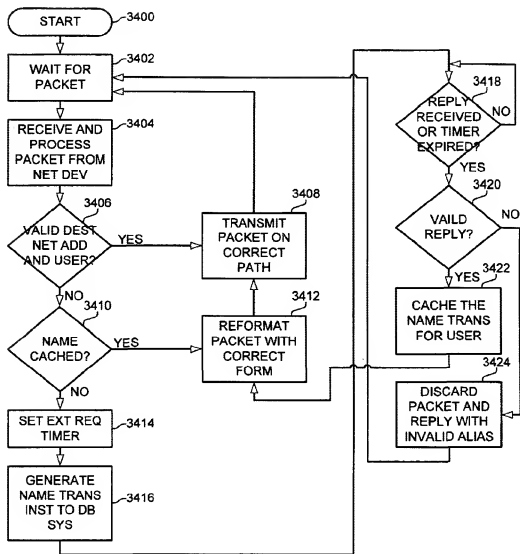


FIG. 34

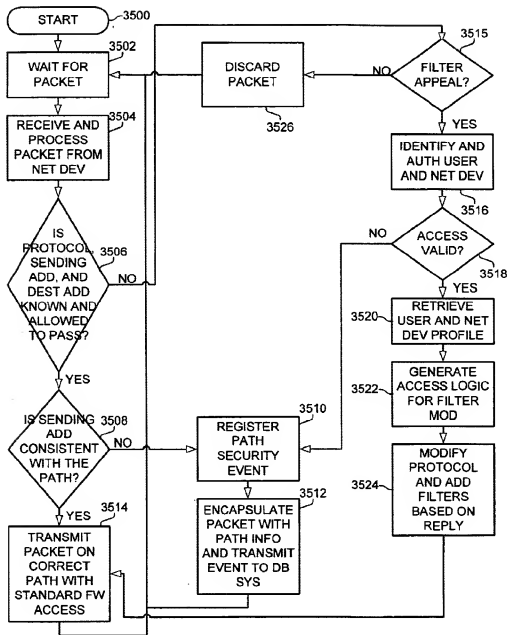


FIG. 35

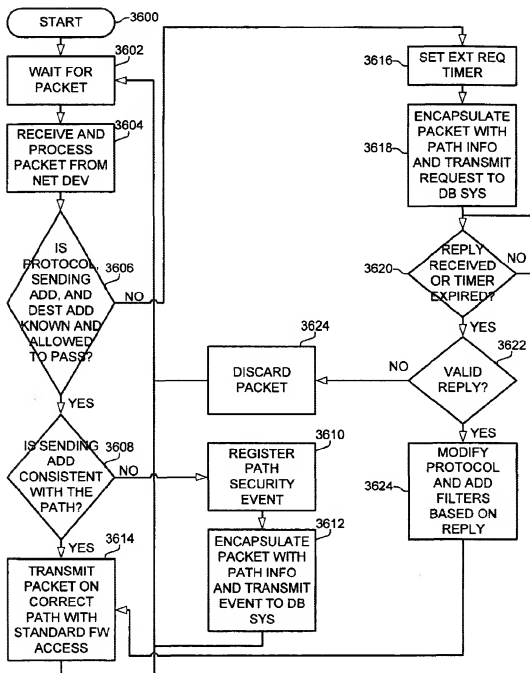


FIG. 36

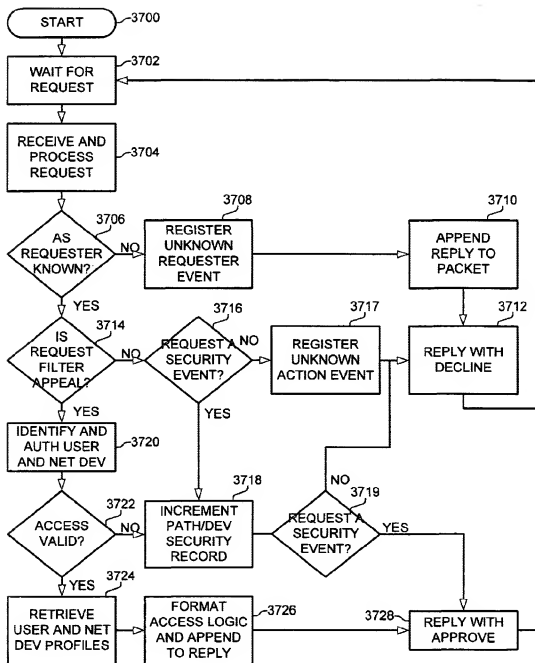


FIG. 37

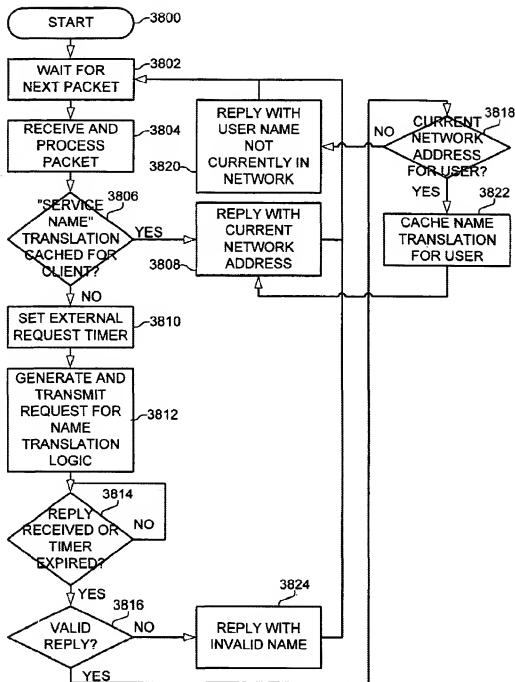


FIG. 38

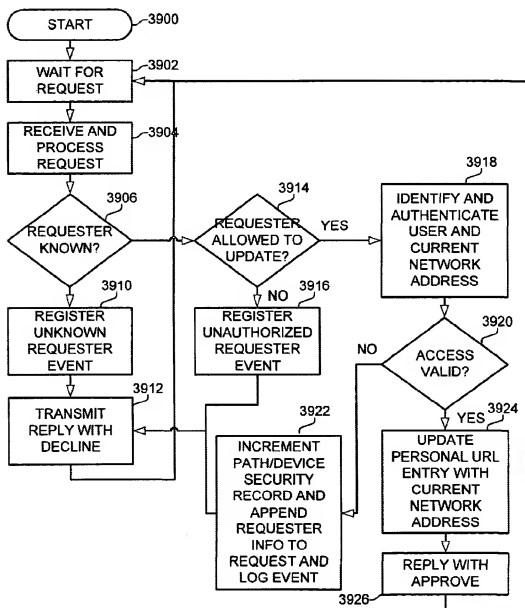


FIG. 39

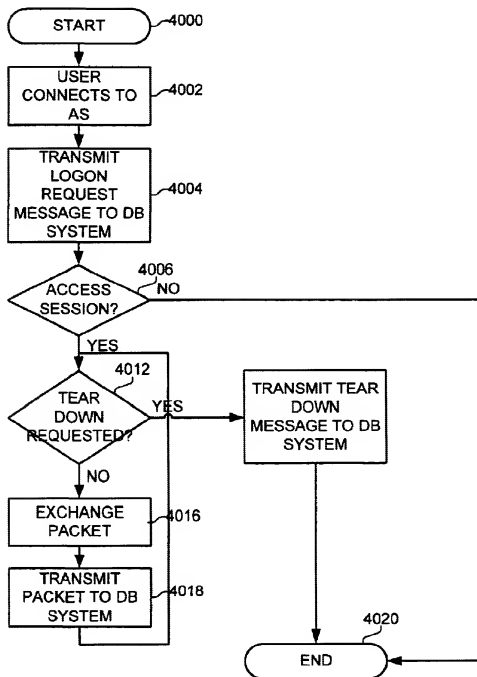


FIG. 40

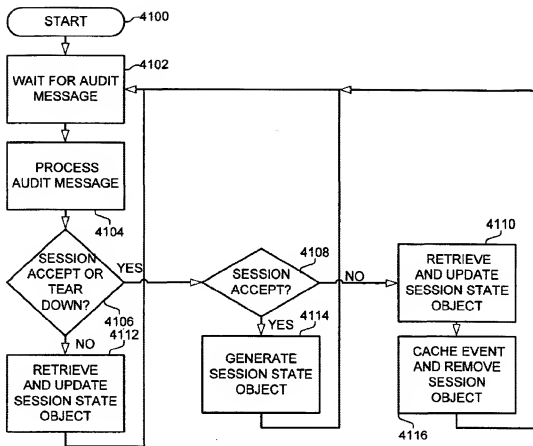


FIG. 41

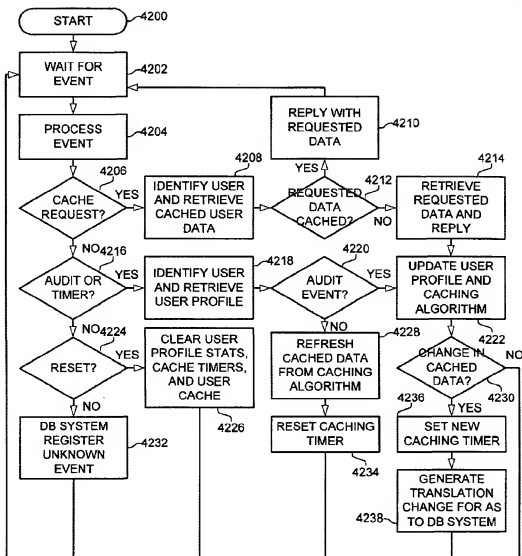


FIG. 42

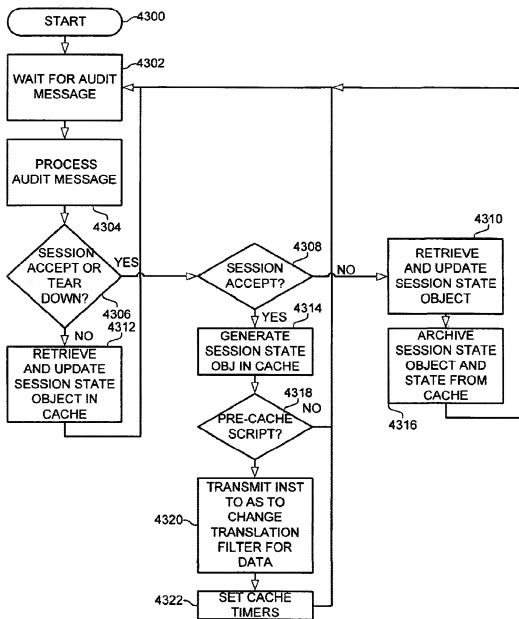


FIG. 43

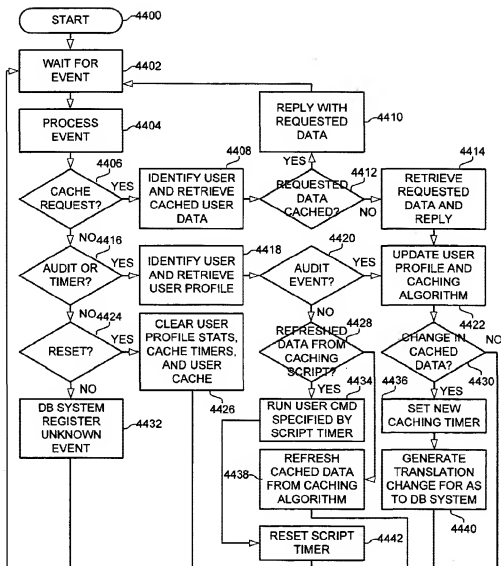


FIG. 44

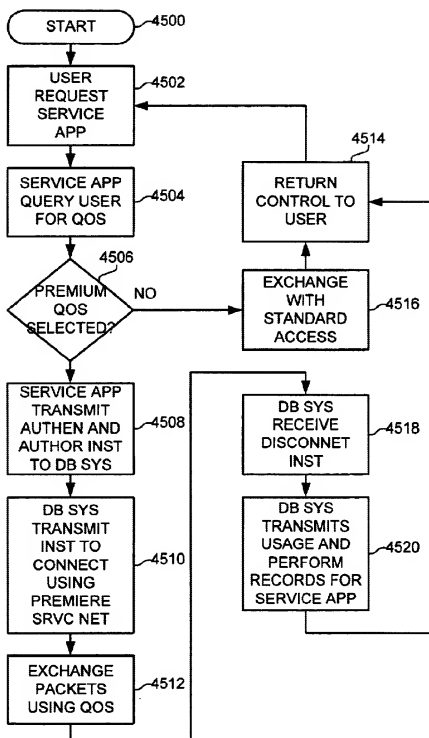


FIG. 45

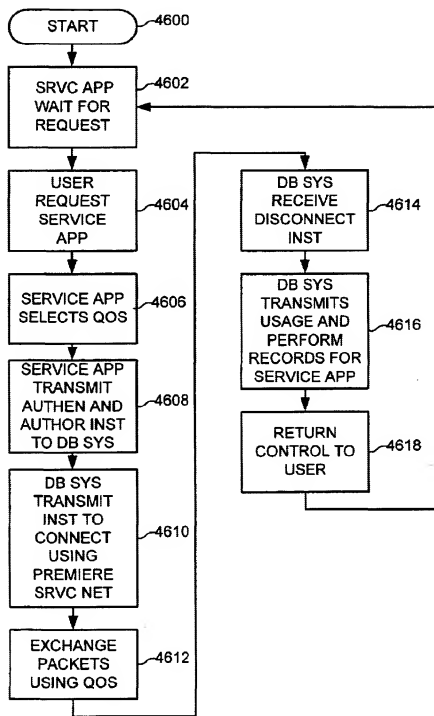


FIG. 46

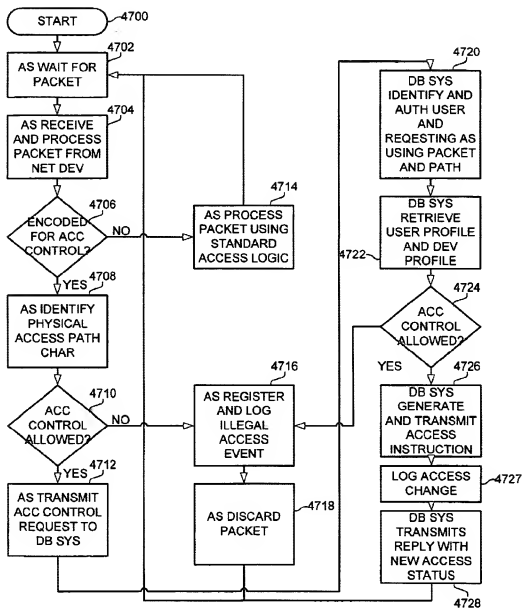


FIG. 47

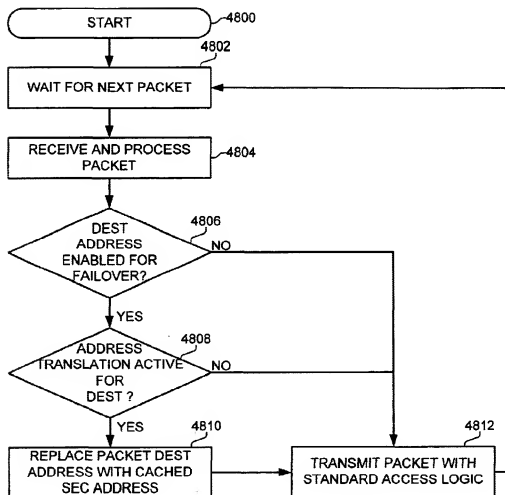


FIG. 48

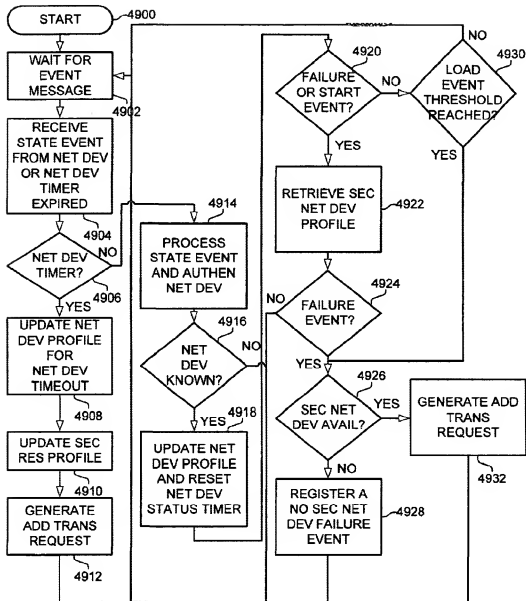


FIG. 49

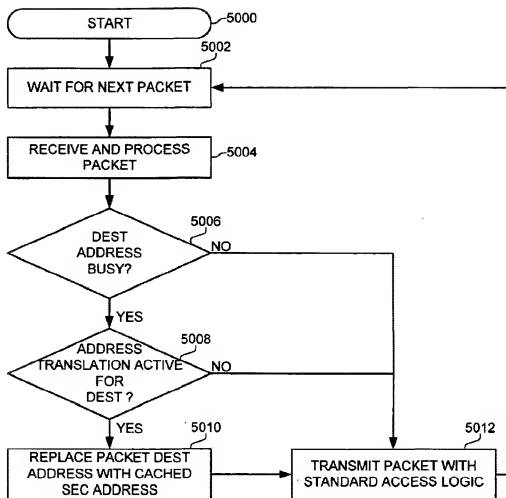


FIG. 50

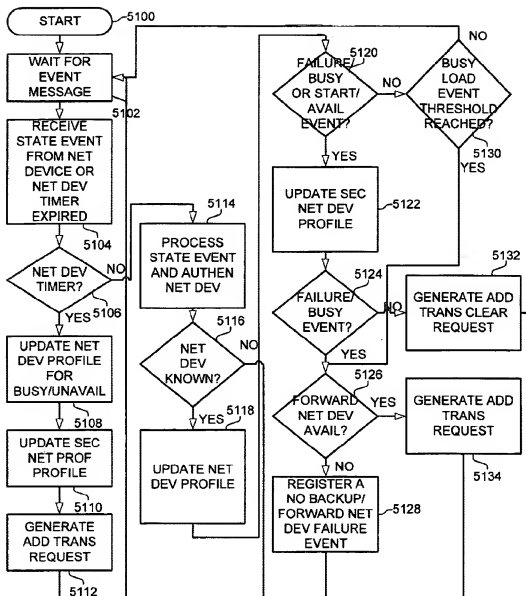


FIG. 51

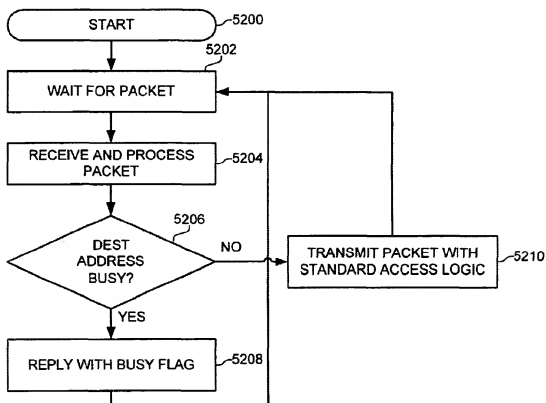


FIG. 52

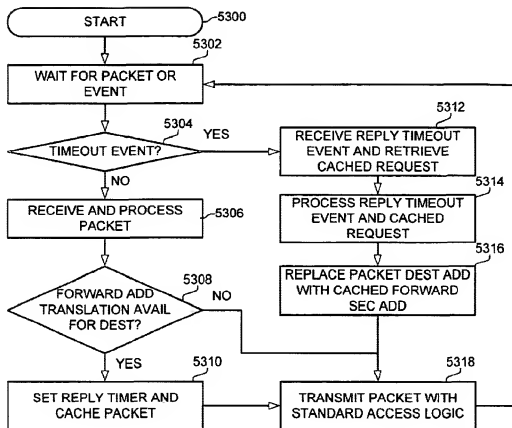


FIG. 53

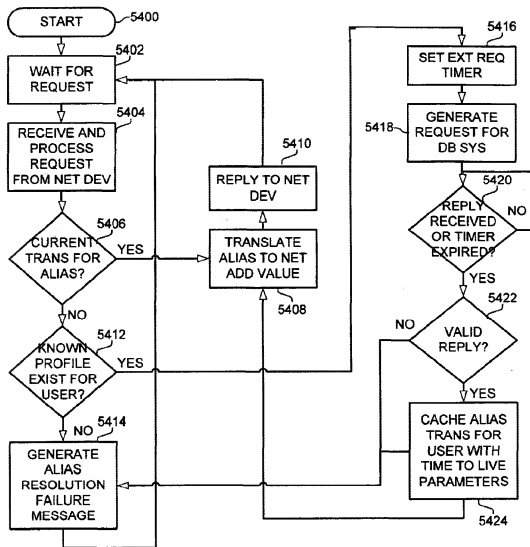


FIG. 54

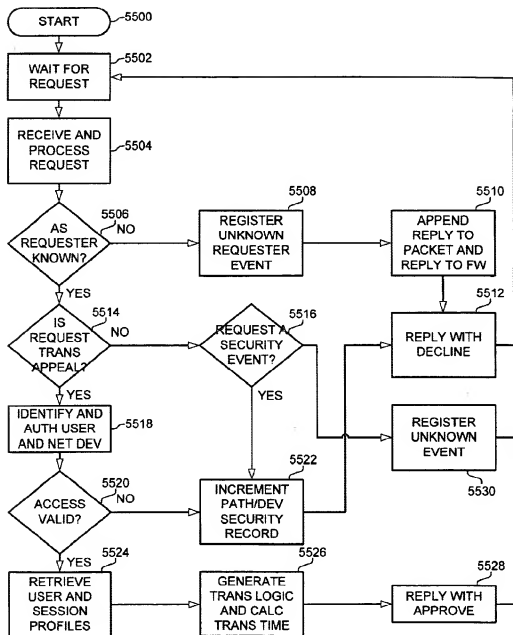


FIG. 55

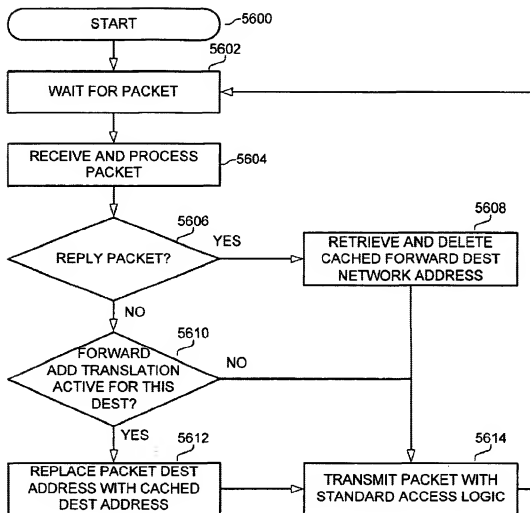


FIG. 56

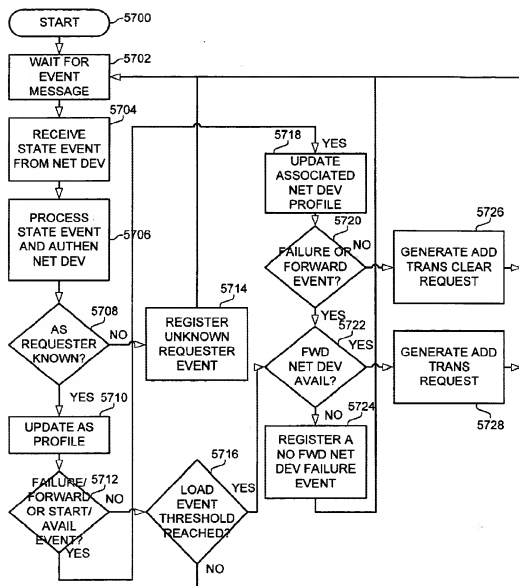


FIG. 57

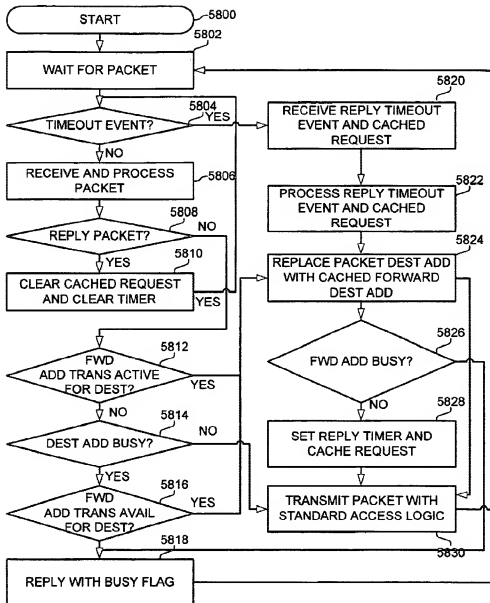


FIG. 58

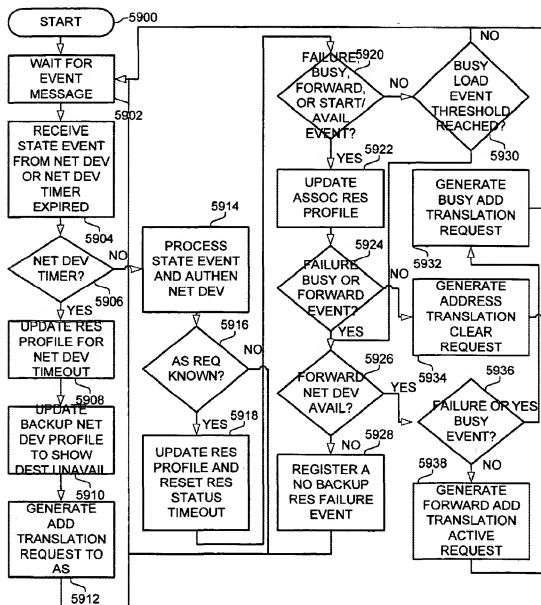
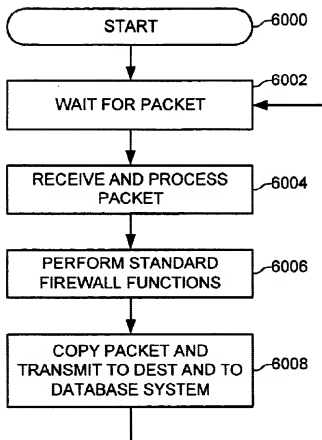


FIG. 59

**FIG. 60**

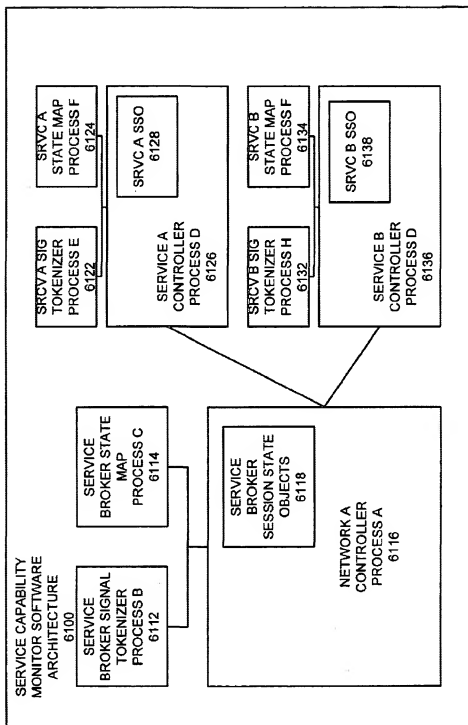


FIG. 61

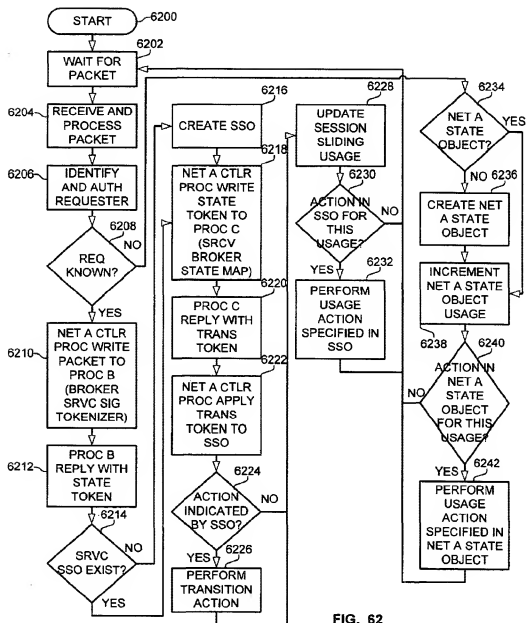


FIG. 62

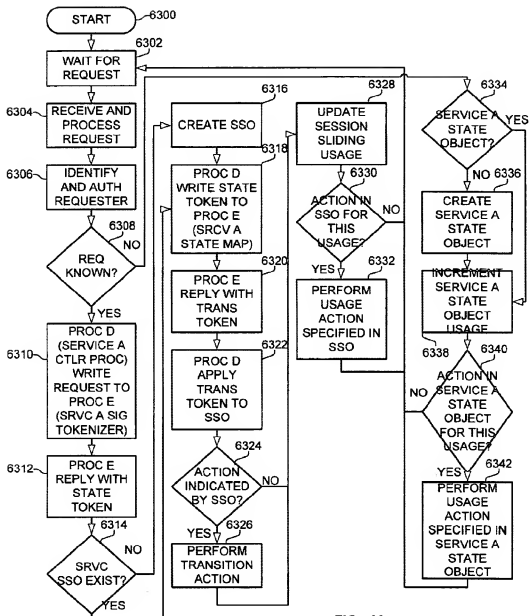


FIG. 63

ACCESS NETWORK AUTHORIZATION

RELATED APPLICATIONS

This application is a continuation of prior application Ser. No. 09/556,276, entitled "Access Communication System", filed Apr. 24, 2000, currently pending, and incorporated by reference into this application.

FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

Not applicable

MICROFICHE APPENDIX

Not applicable

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention is related to the field of communication networks, and in particular, to an access communication system that provides access to multiple service provider systems. More particularly, this invention relates to a system that authorizes access to use the access communication system.

2. Description of the Prior Art

Communication networks have seen dramatic development over the past several years so that today, there are multiple diverse communication networks providing services. The current technical challenge is to develop interfaces between the networks to provide seamless service across multiple networks. Unfortunately, today's interfaces lack the ability to offer the user with easy access to services from multiple systems. These interfaces do not customize operations for the user.

FIG. 1 illustrates the conventional public telephone network. User telephones and computers are connected to local switches. The local switches are coupled to a local database. The user places calls through the local switch. The local switch processes the called number to provide an end-point connection or access to other networks. The local switch may connect the call to another telephone in the local calling area. In a Local Number Portability (LNP) situation, the local switch exchanges information with the local database to obtain the appropriate routing number for a ported call. The local switch may also connect the call to an Internet Service Provider. If a Digital Subscriber Line (DSL) is used, DSL equipment may be used to bypass the local switch. The local switch may also connect the call to a long distance switch. To provide access to the long distance switch, the local switch first exchanges information with the local database. The local database identifies the long distance network for either the user or the dialed number.

The long distance switch processes the called number to route the call to another system or network. Prior to routing, the long distance switch validates the call by checking the caller's number. The long distance switch may also exchange information with the long distance database to provide special call-handling. One example is a calling card call, where the long distance database validates an account number for billing the call. Another example is a toll-free call, where the long distance database processes external information customized by the called party to route the call. Toll-free routing information includes items such as time and date, caller location, and call center status. Many long distance calls are simply routed through the long distance

network to another local network for call completion. Other calls are routed from the long distance network to a call center. Call centers offer a concentration of call-handling capabilities for operations, such as order entry, customer service, and promotions. Call centers include automatic call distribution equipment to route calls to the appropriate destination within the call center.

Both local and long distance networks exchange calls with mobile switches. The mobile switches are connected to base stations that communicate over the air with wireless telephones. When a mobile user places a call, the mobile switches exchange information with a mobile database to validate the mobile caller. The call is then routed to another mobile caller, the telephone network, or to an ISP. When a mobile caller moves around, their wireless phone logs-in with the physically proximate mobile network, and that mobile network updates the mobile user's location in the mobile databases. When a call is placed to that mobile user, their home mobile switch obtains a routing number from the mobile databases to route the call to the mobile network currently in communication with the mobile user.

FIG. 2 illustrates a conventional data network that transfers packets of user data to a destination based on address information carried in the packets. Users are connected to Local Area Networks (LANs) that are connected to Wide Area Networks (WANs). A common LAN is an Ethernet system. A common WAN is an intranet. WANs are interconnected by data networks, such as IP, TI, frame relay, or Asynchronous Transfer Mode (ATM). WANs are connected to the Internet through ISPs. WANs are connected to the public telephone network through telephony gateways. A common telephony gateway is a Private Branch Exchange (PBX).

FIG. 3 illustrates a conventional ISP. The public telephone network is coupled to a telephony interface that converts between telephony analog and digital protocols and the Internet Protocol (IP). Some telephony interfaces also handle DSL traffic that may already use IP. The telephony interface transfers IP traffic through an access server and firewall to a router. Some ISPs combine the firewall and the access server into one system. Also, the position of the firewall may vary, and traffic shapers may be present. The router exchanges IP traffic with the Internet.

In operation, the user calls the ISP over the telephone network and logs-in at the access server. The access server collects and forwards the user name and password to the ISP database. The ISP database validates the user name and password and returns an IP address to the access server. The IP address is for the user's terminal connection. Using the IP address, the user may communicate through the firewall to the router for transmissions to an IP address. The user now has Internet access through the router and exchanges packets with various Internet servers.

IP addresses are referred to as network addresses and include a network ID and a host ID. Network IDs are unique across the Internet and host IDs are unique within a given network. IP addresses are lengthy numerical codes, so to simplify things for the user, service addresses are available that are easier to remember. The service addresses are often the name of the business followed by ".com". Domain Name Service (DNS) is hosted by servers on the Internet and translate between service addresses and network addresses. The browser in the user computer accesses the DNS to obtain the desired network address.

FIG. 4 illustrates conventional network access. A current proposal for communication network access is provided by

3

the Telecommunication Information Network Architecture Consortium (TINA-C). TINA-C proposes the use of agents in the user domain and the service provider domain. The service provider domain could be a telephone network, data network, or ISP. The agents negotiate access service rights. Once the service is negotiated, the user receives the service from the service provider network during a service session. Unfortunately, the access session occurs between the user domain and a particular service provider domain. At present, the service provider domain provides limited access capability beyond simply handing off communications to another network based on a called number or network address. As a result, the ability to customize services for a particular user across multiple service providers is inadequate.

SUMMARY OF THE INVENTION

The inventions solve the above problems by providing access between a user system and a plurality of communication networks. The plurality of communication networks provide services to a user in the user system. An access communication system includes a database system and an access server that is connected to the user system and the plurality of communication networks.

In one aspect of the inventions for user access profile inheritance, the database system receives an update request from the access server to update a user access profile through inheritance. The database system then processes the update request to inherit user profile information from a user profile data structure. The database system updates the user access profile with the user profile information.

In another aspect of the inventions for network shells, the access server receives an alias selection from a user for a network shell that includes alias selections associated with actions. The access server then processes the alias selection to execute an action associated with the alias selection.

In another aspect of the inventions for service based directory, the access server transmits a list of services to a user system. The access server then receives a selection from the list of services. The access server processes the selection to generate an instruction to provide the service related to the selection.

In another aspect of the inventions for user access profile mobility, the database system receives user information. The database system then processes the user information to determine if a user access profile is local within a local database system. The database system generates and transmits a request to retrieve a user access profile from a second database system external to the local database system in response to the determination that the user access profile is not local.

In another aspect of the inventions for service, user, and device sessions, the access communication system establishes a connection between a network device and the access server. The access communication system then generates a device session including a device session ID based on the network device. The access communication system generates and transmits a login query for the network device. The access communication system receives and processes a login reply from the network device to generate a user session including a user session ID based on the user. The access communication system receives and processes a request for the service to generate a service session including a service session ID based on the service. The service may generate and transmit a login query for the user. The access communication system links the device session, user session, and the service session using the device session ID, the user session ID, and the service session ID.

4

In another aspect of the inventions for service capability firewall, the access server receives information including a named function request for a service provider. The access server processes the information to check if the named function request is valid for the service provider and the service. If valid, the access server determines if a private destination address exists for the named function request. The access server replaces the named function request with the private destination address in response to the determination that the private destination address exists for the named function request. The access server then transmits the information with the private destination address to the service provider.

In another aspect of the inventions for prepaid access and bank card access, the database system receives information identifying a billing code for a user. The database system then processes the billing code to determine if the user is allowed to use the access system. The database system provides access to the access system in response to the determination that the user is allowed to use the access system.

In another aspect of the inventions for global authentication and access card, the database system receives a user login. The database system then processes the user login to determine if the user is allowed access to the access communication system based on a local database system. The database system then provides access to the access communication system to the user in response to the determination that the user is allowed access based on the local database system. The database system then generates an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system. The database system receives and processes an authorization response indicating whether the user is allowed to use the access system from the second database system. The database system then provides access to the access communication system to the user in response to the authorization response that allows the user to use the access communication system.

In another aspect of the inventions for user based proxy and subscriber based proxy, the database system includes a user proxy. The user proxy receives a request for the service from the user system. The user proxy then transmits the request for the service to a service provider. The user proxy exchanges user information between the user system and the service provider.

In another aspect of the inventions for dynamic proxy, the database system includes a proxy. The proxy receives a service/protocol request for a new service or protocol. The proxy processes the service/protocol request to generate a handler request to obtain a handler for the new service or protocol. The proxy then receives and executes the handler for the new service or protocol.

In another aspect of the inventions for access execution environment, the database system receives and processes a login reply into an access execution environment for a user. The database system retrieves programs for the user into the access execution environment. The database system executes the programs for the user in the access execution environment.

In another aspect of the inventions for domain name scoping and inband domain name service lookup, the access server receives information including an alias from the user system. The access server determines if the alias exists in a cache including aliases and alias translations for the user.

5

The access server changes the information based on the cached alias translation.

In another aspect of the inventions for inline access service triggering, the access server receives information. The access server then processes the information to determine if the information is allowed to pass. The access server changes access logic based on the information in response to the determination that the information is not allowed to pass. The access server changes the filters of the access server based on the information in response to the determination that the information is not allowed to pass.

In another aspect of the inventions for access service triggering, the access server receives information. The access server processes the information to determine if the information is allowed to pass. The access server then generates a request from a database system in response to the determination that the information is not allowed to pass. The access server receives a reply including access logic from the database system. The access server changes filters of the access server based on the access logic.

In another aspect of the inventions for personal URL, the database system receives information including a user alias. The database system processes the information to determine if a user alias translation including a current network address for the user alias exists. The database system then modifies the information with the current network address using the user alias translation.

In another aspect of the inventions for predictive caching, the access server receives a request for data. The access server then determines if the data exists in a user cache wherein the user cache contains cached data based on the user's predictive patterns. The access server retrieves the data from the user cache in response to the determination that the data exists in the user cache. The access server transmits the request for data for the service provider in response to the determination that the data does not exist in the user cache.

In another aspect of the inventions for user controlled caching, the access server receives a request for data. The access server determines if the data exists in a user cache wherein the user cache contains cached data based on a user's script of commands. The access server retrieves the data from the user cache in response to the determination that the data exists in the user cache. The access server then transmits the request for data for the service provider in response to the determination that the data does not exist in the user cache.

In another aspect of the inventions for service usage audit, the access server receives an audit message into a database system. The access server processes the audit message to store the audit message in the database system.

In another aspect of the inventions for switching access by a user, switching access by a service provider, and dynamic access control, the database system receives a request. The database system processes the request to determine if the switching of the access is allowed. The database system then generates an instruction to switch access in response to the determination that the switching is allowed.

In another aspect of the inventions for network failover, network busy forwarding, time-out, busy flag, forwarding, and network endpoint availability management, the access server receives information for a destination network device. The access server determines if the destination network device is available. The access server performs an action in response to the determination that the destination network device is unavailable.

6

In another aspect of the inventions for scheduled alias translation, the access server receives information including an alias. The access server processes the information to determine whether an alias translation exists based on an alias translation schedule. The access server then modifies the information based on the alias translation in response to the determination the alias translation exists.

In another aspect of the inventions for service capability monitor, the database system receives information from an access server during a service session. The database system determines a current state of the service session based on the information. The database system determines a state transition based on the current state and a map of state transitions of the service. The database system determines whether the state transition is valid for the service session.

BRIEF DESCRIPTION OF THE DRAWINGS

The same reference number represents the same element on all drawings.

FIG. 1 illustrates a public telephone network in the prior art.

FIG. 2 illustrates a data network in the prior art.

FIG. 3 illustrates an Internet Service Provider in the prior art.

FIG. 4 illustrates conventional network access.

FIG. 5 illustrates a network architecture in an example of the invention.

FIG. 6 illustrates an access network in an example of the invention.

FIG. 7 illustrates a table for a user access profile in an example of the invention.

FIG. 8 illustrates a flowchart for an access server for inheriting a user access profile in an example of the invention.

FIG. 9 illustrates a flowchart for a database system for inheriting a user access profile in an example of the invention.

FIG. 10 illustrates a flowchart of an access server for executing a network shell in an example of the invention.

FIG. 11 illustrates a flowchart of a database system for updating a network shell in an example of the invention.

FIG. 12 illustrates a flowchart for the services based directory in an example of the invention.

FIG. 13 illustrates a flowchart of user access profile mobility in an example of the invention.

FIG. 14 illustrates a logical view of device, user, and service sessions in an example of the invention.

FIG. 15 illustrates a flowchart for a user based session in an example of the invention.

FIG. 16 illustrates a flowchart for a service based session in an example of the invention.

FIG. 17 illustrates a flowchart for a firewall/router for service capability firewall in an example of the invention.

FIG. 18 illustrates a flowchart for a database system for service capability firewall in an example of the invention.

FIG. 19 illustrates a flowchart for prepaid access in an example of the invention.

FIG. 20 illustrates a flowchart for bank card access for a connection in an example of the invention.

FIG. 21 illustrates a flowchart for network access cards for a disconnection in an example of the invention.

FIG. 22 illustrates a flowchart for network access cards for a connection in an example of the invention.

7

FIG. 23 illustrates a flow chart for network access cards for a disconnection in an example of the invention.

FIG. 24 illustrates a flowchart for global access in an example of the invention.

FIG. 25 illustrates a flowchart for an access server for user based proxies in an example of the invention.

FIG. 26 illustrates a flowchart for a user proxy for user based proxies in an example of the invention.

FIG. 27 illustrates a flowchart for an access server for subscriber based proxies in an example of the invention.

FIG. 28 illustrates a flowchart for a subscriber proxy for subscriber based proxies in an example of the invention.

FIG. 29 illustrates a flowchart for a dynamic proxy for dynamic proxies in an example of the invention.

FIG. 30 illustrates a block diagram of the access execution environment in an example of the invention.

FIG. 31 illustrates a flow chart for the access execution environment in an example of the invention.

FIG. 32 illustrates a flowchart for an access server for domain name scoping in an example of the invention.

FIG. 33 illustrates a flowchart for a database system for domain name scoping in an example of the invention.

FIG. 34 illustrates a flowchart for an access server for an inband domain name service lookup in an example of the invention.

FIG. 35 illustrates a flowchart for inline access service triggering in an example of the invention.

FIG. 36 illustrates a flowchart for an access server for access service triggering in an example of the invention.

FIG. 37 illustrates a flowchart for a database system for access service triggering in an example of the invention.

FIG. 38 illustrates a flow chart for the personal URL lookup in an example of the invention.

FIG. 39 illustrates a flow chart for the personal URL update in an example of the invention.

FIG. 40 illustrates a flowchart for an access server for auditing in an example of the invention.

FIG. 41 illustrates a flowchart for a database system for auditing in user predictive caching in an example of the invention.

FIG. 42 illustrates a flowchart for a database system for caching in user predictive caching in an example of the invention.

FIG. 43 illustrates a flowchart for a database system for auditing in user controlled caching in an example of the invention.

FIG. 44 illustrates a flowchart for a database system for caching in user controlled caching in an example of the invention.

FIG. 45 illustrates a flowchart for switching access by a user in an example of the invention.

FIG. 46 illustrates a flowchart for switching access by a service provider in an example of the invention.

FIG. 47 illustrates a flowchart for dynamic switching access in an example of the invention.

FIG. 48 illustrates a flowchart for an access server for network address failover in an example of the invention.

FIG. 49 illustrates a flowchart for a database system for network address failover in an example of the invention.

FIG. 50 illustrates a flowchart for an access server for network busy forwarding in an example of the invention.

FIG. 51 illustrates a flowchart for a database system for network busy forwarding in an example of the invention.

8

FIG. 52 illustrates a flowchart for an access server for a busy flag when the destination network device is busy in an example of the invention.

FIG. 53 illustrates a flowchart for an access server for forwarding, if the destination network device timeouts in an example of the invention.

FIG. 54 illustrates a flowchart for an access server for schedule alias resolution in an example of the invention.

FIG. 55 illustrates a flowchart for a database system for scheduled alias resolution in an example of the invention.

FIG. 56 illustrates a flowchart for an access server for destination controlled forwarding in an example of the invention.

FIG. 57 illustrates a flowchart for a database system for destination controlled forwarding in an example of the invention.

FIG. 58 illustrates a flowchart for an access server for network endpoint availability management in an example of the invention.

FIG. 59 illustrates a flowchart for a database system for network endpoint availability management in an example of the invention.

FIG. 60 illustrates a flowchart for a firewall/router for service capability monitor in an example of the invention.

FIG. 61 illustrates a service capability monitor software architecture for a service capability monitor in an example of the invention.

FIG. 62 illustrates a flowchart for the network logic for service capability monitor in an example of the invention.

FIG. 63 illustrates a flowchart for the service logic for service capability monitor in an example of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Communication Network Architecture—FIGS. 5 and 6

The following description and associated figures discuss specific examples intended to teach the present invention to those skilled in the art. Those skilled in the art will appreciate numerous variations from these examples that do not depart from the scope of the invention. Those skilled in the art will also appreciate that various features described below could be combined in various ways to form multiple variations of the invention.

FIG. 5 illustrates a network architecture 500 in an example of the invention. The network architecture 500 comprises a service network 530, a service network 540, and an access network 520. The access network 520 comprises a database system 522, an access server 524, a firewall/router 526, a firewall/router 556, and an access server 554. A user network 510 includes a network device 512. The user network 510 is connected to the access server 524. The access server 524 is connected to the firewall/router 526 and the database system 522. The firewall/router 526 is connected to the service network 530, the service network 540, and the database system 522. The firewall/router 556 is connected to the service network 530, the service network 540, the database system 522, and the access server 554. The access server 554 is connected to the database system 522 and the user network 560. The user network 560 comprises a network device 562 and a network device 564.

The access network 520 provides an interface between the user network 510 and 560 and the service networks 530 and

540. The interface function provides user access profiles, security, switching, and caching. The user network 510 and 560 could be a residential or business communication system that includes network devices. A network device 512, 562, and 564 could be any device configured to exchange data or information with the access network 520. Some examples of network devices are wireless and wireline telephones, computers, modems, servers, and/or data terminals, along with associated interconnections and network interfaces. For illustrative purposes in the examples below, a user interacts with the network device 512 to access services provided by the network architecture 500 and the user network 560. The user network 510 could also be a communication destination for other users. In these cases, the access network 520 provides destination performance monitoring and control. The example of a user interacting with the network device 562 and 564 to access services provided by the network architecture 500 and the user network 510 is not discussed below for the sake of clarity.

The access server 524 and the database system 522 could be adapted from components used in current ISPs, and the access network 520 could be integrated into these ISP components. The database system 522 houses user access profiles for the users in the user networks 510 and 560 with external access rights. The user access profiles provide a data clearinghouse for user-related information for security, service options, current state, and customized macros. The service networks 530 and 540 could be voice or data systems, such as the public telephone network, Internet, public data networks, and private data networks.

When a user requests access to services, the access network 520 processes the user access profile for the user. The access network 520 performs security measures to validate the user. The access network 520 then binds the user to a terminal and to a service. This user-terminal-service binding correlates the user's capabilities in the user access profile with the capabilities of the user's current terminal and the current capabilities of the service provider. The access network 520 activates a network shell for the user and the user's terminal. The network shell is an interface that is customized for the user and the terminal. The user invokes services using the network shell.

The user access profile may contain several macros that can be as simple as address translations or as complex as lengthy computer programs. The user's network shell provides access to the user's macros. The user access profile also stores caching instructions. The caching instructions control the collection and storage of information within the access network 520 for immediate access by the user.

FIG. 6 depicts an access network in an example of the invention. The access network 520 comprises the database system 522, the access server 524, the firewall/router 526, the firewall/router 556, and the access server 554. The database system 522 comprises a local database system 570, a Lightweight Directory Access Protocol (LDAP) interface system 571, a central database system 580, a local database system 590, and an LDAP interface system 591. The local database system comprises a user profile system 572, an audit database system 573, a cache database system 574, a host system 575, a security server 576, a user authorization system 577, an alias translation system 578, and a personal DNS system 579. The central database system 580 comprises a user authorization system 581, a financial interface 582, and a cross connect system 583. The local database system 590 comprises a user profile system 592, a user authorization system 593, and an availability system 594.

The access server 524 and the firewall/router 526 are connected to the local database system 570. The local

database system 570 is connected to the central database system 580 and the LDAP interface system 571. The central database system 580 is connected to the local database system 590. The local database system 590 is connected to the firewall/router 556, the access server 554, and the LDAP interface system 591.

An access provider is an entity that provides access to users who use communications services. A typical access provider comprises an access server, a firewall/router, and a local database system. The systems within the local database system 570 could be included within the local database system 590. Also, the systems within the local database system 590 could be included within the local database system 570. As discussed above, the user is accessing the network architecture through the user network 510. The duplication of systems in the local database system 570 and the local database system 590 is excluded for the purposes of clarity. A service provider is an entity that provides communication services to users who are accessing the service through an access provider.

Network User Access Profile—FIGS. 5, 6, 7, 8, 9, 10, 11, and 12

User Access Profile Inheritance

The user access profile is stored in the access network 520 and controls user access to services. The user access profile is any information or data associated with controlling user access to a service such as role identification, authorization, billing information and access preferences.

FIG. 7 depicts table for a user access profile in an example of the invention. A user access profile includes access information for the user, billing information, and preferences for access. Access information is any information or data related to providing the user access to the network architecture 500. Some examples of access information are user ID, password, name, account number, user alias, current network address, switching allowed flag, and other security information. Access information also include a list of services that the user has subscribed to or is allowed access to. In one embodiment, access information includes a cache of information that the user has accessed previously. Some examples of billing information are address and billing code including bank card numbers or prepaid account codes. Preferences for access allow the user to save choices or preferences to customize their access to the network architecture 400. Some examples of preferences for access are file formats and Quality of Service values. The user access profile may also include usage information such as time of day access, day of week, usages per day, usages per week, and usages per month.

Users typically set up their user access profiles when signing up for the access to the network architecture 500. In one embodiment, users create the user access profile through an inheritance process. Through the inheritance process, the user selects user profile information from other user access profiles in the network architecture 500. The user may inherit user profile information from user profile data structures such as templates, other user access profiles, the user's group or class, or other networks that the user is set up in. A user profile data structure is any user profile information to be retrieved when inheriting a user access profile.

In a prior solution, a command interpreter in a single computer operating system shell environment allows a user to customize the interpretation of command strings. The user may inherit portions of other user's shell environment by adding the other user's shell attributes. User access profiles are typically stored in the access provider's database. FIGS.

5 and FIG. 8 disclose one embodiment for inheriting user access profiles in an example of the invention. The user access profile is created through an inheritance process where the user is able to select capabilities, macros, functions, methods, and data to inherit from other profiles. In this embodiment, users inherit user access profiles from classes, groups, or provider recommendations. Features of user access profiles such as alias translation could then be implemented with new users rapidly. The inheritance of user access profiles also simplifies the configuration of users. In one example, a user initiates the user access profile inheritance by clicking a button on a website. Also, when a user requests a new service, the service provider automatically inherits the user access profile for the user so the user is able to use the requested service.

FIG. 8 depicts a flowchart for the access server 524 for inheriting a user access profile in an example of the invention. FIG. 8 begins in step 800. In step 802, the access server 524 waits for the next packet. The access server 524 then receives and processes the packet from the network device 562 in step 804. The access server 524 then checks if the destination is the user access profile in step 806. If the destination is not the user access profile, the access server 524 transmits the packet on the correct path in step 808 before returning to step 802. If the destination is the user access profile, the access server 524 then checks if the network device 562 is allowed access to the user access profile in step 1110. If the network device 512 is not allowed access to the user access profile, the access server 524 discards the packet and registers a network request security event in step 812 before returning to step 802.

If the network device 512 is allowed access to the user access profile, the access server 524 then checks if the user is allowed to update their user access profile in step 814. If the user is not allowed updates to their user access profile, the access server 524 generates and transmits a profile update not allowed message to the network device 512 in step 816 before returning to step 802. If the user is allowed updates to their user access profile, the access server 524 generates and transmits a user access profile update permission message asking if the user wishes to update the user access profile to the network device 512 in step 818. The access server 524 then checks if the user approved the user access profile update in step 820. If the user does not approve, the access server 524 generates and transmits a user access profile aborted message to the network device 512 in step 822 before returning to step 802.

If the user approves the user access profile update, the access server 524 sets an external request timer in step 824. The access server 524 then generates and transmits a user access profile update request to the database system 522 in step 826. A user access profile update request could be any signaling, message, or indication to update the user access profile. The access server 524 then checks if a reply was received or the external request timer expired in step 828. If the reply was not received and the external request timer did not expire, the access server 524 returns to step 828. If the reply was received or the external request timer did expire, the access server 524 checks if the reply was valid in step 830. If the reply was valid, the access server 524 generates an update complete message in step 832 before returning to step 802. If the reply was not valid, the access server 524 discards the packet and replies to the network device 512 that the user access profile update failed in step 834 before returning to step 802.

FIG. 9 depicts a flow chart for the database system 522 for inheriting a user access profile in an example of the inven-

tion. FIG. 9 begins in step 900. In step 902, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 904. In step 906, the database system 522 then checks if the access server requester is known. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 912. The database system 522 appends the reply to the packet and transmits the reply to the access server 524 in step 910. The database system 522 then replies with a decline message to the access server 524 in step 912 before returning to step 902.

If the requester is known, the database system 522 then checks whether the request is a user access profile update request in step 914. If the request is not a user access profile update request, the database system 522 checks if the request is a security event in step 916. If the request is not a security event, the database system 522 then registers an unknown action event in step 918 before returning to step 910. If the request is a security event, the database system 522 increments a path/device security record in step 924. The database system 522 then logs the path/device security record in step 926 before returning to step 902.

If the request is a user access profile update request, the database system 522 identifies and authenticates the user and the network device 512 in step 920. The database system 522 then checks if the access is valid in step 922. If the access is invalid, the database system 522 returns to step 924. If the access is valid, the database system 522 retrieves the user access profile information from a user profile data structure in step 928. The retrieval of user access profile information may be based on any information in the user access profile such as group or class or selections of inheritance user access profile presented to the user. The group or classes could be any logical grouping of user based on similar interests or situations such as work, family, and geographic location. The database system 522 then updates the user access profile in step 930 based on the user access profile information retrieved in step 928 and the user's selections for updating. The database system 522 then replies with an approve message to the access server 524 in step 932 before returning to step 902. In one embodiment of the invention, the database system 522 uses the user access profile system 572 to perform the operations disclosed in FIG. 9 and stores the user access profile in the user access profile system 572.

45 Network Shell

The user access profile includes data to provide a customized network shell to the user. The network shell is a user interface to the access network that is linked to programs, methods, macros, or service. Typically, configuration of the network shell is graphically presented to the user on a display. For example, the network shell appears as a list of alias selections that are associated with actions such as a program or macro for resources or services. An alias selection is any information that is associated with an action to be executed when the alias selection is selected. In another example, the alias selections are graphically presented as icons that relate to actions to be executed. The network shell overlays the standard DNS offered in IP networks, which reduces alias translation delays and required user keystrokes. In prior solutions, a command interpreter in an operating system shell environment allows a user to customize the interpretation of command strings, or a DNS translates "alias" values to addresses. FIGS. 5, 10, and 11 show one embodiment for network shells in an example of the invention.

FIG. 10 depicts a flowchart of an access server for executing a network shell in an example of the invention.

13

FIG. 10 begins in step 1000. In step 1002, the access server 524 waits for the next packet. The user selects an alias selection from the network shell graphically presented. In another embodiment, the user enters an alias value by a non-graphical means. Alternatively, the network shell is not presented to the user. The network device 512 transmits a packet including the alias selection to the access server 524. In step 1004, the access server 524 receives and processes the packet from the network device 512 and processes the packet. The access server 524 then checks if the network device 512 is allowed access to the network architecture 500. If the network device 512 is not allowed access to the network architecture 500, the access server 524 discards the packet and registers a network security event in step 1008 before returning to step 1002.

If the network device 512 is allowed access to the network architecture 500, the access server 524 checks if the user is recognized in step 1010. If the user is not recognized, the access server 524 returns to step 1008. If the user is recognized, the access server 524 retrieves the user's network shell in step 1012. In some embodiments, the user's network shell is retrieved from the user access profile in the access database 522 prior to presenting the network shell to the user. In step 1014, the access server 524 checks if the packet from the network device 512 includes an alias selection from the user's network shell. In one embodiment, specialized hardware is used to scan for aliases just as IP addresses are currently scanned for. If the packet does not include an alias selection from the user's network shell, the access server 524 proceeds to step 1018. If the packet does include the alias selection from the user's network shell, the access server 524 executes the action associated with the alias selection in step 1016 before returning to step 1002. In step 1018, the access server 524 processes the packet with the normal handling before returning to step 1002.

FIG. 11 depicts a flowchart of a database system for updating a network shell in an example of the invention. FIG. 11 begins in step 1120. In step 1122, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 1124. The database system 522 then checks if the access server requester is known in step 1126. If the access server requester is not known, the database system 522 registers an unknown requester event in step 1128. The database system 522 then appends the reply to the packet and transmits the packet to the access server 524 in step 1130. In step 1132, the database system 522 replies with a decline message to the access server 524 before returning to step 1122.

If the access server requester is known, the database system 522 checks if the request is a profile update for the network shell in step 1134. If the request is a profile update for the network shell, the database system 522 identifies and authenticates the user and the network device 512 in step 1136. The database system 522 then checks whether the access is valid in step 1138. If the access is invalid, the database system proceeds to step 1154. If the access is valid, the database system 522 retrieves the user access profile in step 1140. The database system 522 then updates the user access profile with the network shell with the user's alias selections and the associated programs, macros, functions or methods in step 1142. The database system 522 then replies with an approve message in step 1144 before returning to step 1122.

If the request is not a profile update, the database system 522 checks if the request is an action event in step 1146. If the request is an action event, the database system 522

14

performs the action and replies in step 1148 before returning to step 1122. If the request is not an action event, the database system 522 checks if the request is a security event in step 1150. If the request is not a security event, the database system 522 registers an unknown request type event in step 1152 before returning to step 1122. If the request is a security event, the database system 522 proceeds to step 1154. In step 1154, the database system increments a path/device security record. The database system 522 then appends the requester information to the packet and logs the event before returning to step 1122. In one embodiment of the invention, the database system 522 uses the user access profile system 572 to perform the operations disclosed in FIG. 11 and stores the user access profile in the user access profile system 572.

Service Based Directory

In prior systems, the RADIUS server provided the user with selections for transport modes. However, the user was not able to select available network based services. FIGS. 5 and 12 disclose one embodiment for a service based directory in an example of the invention. In this embodiment, access providers provide a list of services that the user can select. With the list of services, the access providers have the ability to advertise specific services to users. The list of services may be generated based on the user access profile to make the list user specific. Once the user makes the selection, the access server 524 connects the user to the service network such as Intranet, Internet, or private dedicated network that provides the selected service.

FIG. 12 depicts a flowchart for the services based directory in an example of the invention. FIG. 12 begins in step 1200. In step 1202, the access server 524 waits for the next connection from a network device in the user network 510. The access server 524 then receives a connection from the network device 512 in step 1204. Once the connection is established, the access server 524 generates and transmits a user ID query to the network device 512 in step 1206. The access server 524 then receives an ID reply and establishes a network device session in step 1208.

The access server 524 then generates an available services reply including a list of services in 1210. In one embodiment, the access server 524 generates the available services reply based upon information in the user access profile. The access server 524 receives a selected service reply from the network device 512 in step 1212. The access server 524 then connects the network device 512 to the selected service provider in step 1214. The access server 524 waits for the next packet in step 1214. The access server 524 then exchanges packets between the network device 524 and the selected service provider in step 1218.

Access Network User Binding—FIGS. 5, 13, 14, 15, and 16

User Access Profile Mobility

Users may access their user access profile from any network device connected to the network architecture 500. In a prior solution, a command interpreter in an operating system shell environment allows a user to customize the interpretation of command strings. In this environment, the access network identifies the user and the terminal device interface, which provides user mobility. The access network then executes customized actions for the user. FIGS. 5 and 13 show one embodiment of the invention for user access profile mobility. This embodiment provides user mobility in a distributed data network environment.

FIG. 13 depicts a flowchart of user access profile mobility in an example of the invention. FIG. 13 begins at step 1300. A user at the network device 512 signs on the access network

15

520 with their user ID. The network device 512 transmits user information with the user ID to the access server 524. In this embodiment, the user information is in the form of a packet. In step 1302, the access server 524 receives and processes the packet to check if the user access profile is local within the local database system 570. A user access profile is local within the local database system 570 when the user access profile is located in the local database system 570. If the user access profile is local, the access server 524 proceeds to step 1312.

If the user access profile is not local within the local database system 570, the access server 524 checks if the user ID is delimited with a provider ID in step 1304. One example of a user ID delimited with a provider ID includes a user's name and a provider ID separated by a delimiter such as joesmith@access.net. If the user ID is not delimited, the access server 524 retrieves the location of the default user access profile system using a default Lightweight Directory Access Protocol (LDAP) interface system 571 in step 1308 before proceeding to step 1312.

If the user ID is delimited, the access server 524 checks if the provider ID is valid in step 1306. If the provider ID is not valid, the access server 524 registers an invalid profile request security event before terminating at step 1326. If the provider ID is valid, the access server 524 uses the provider ID to retrieve the location of the local database system 590 from a foreign LDAP interface system 591 before proceeding to step 1312.

In step 1312, the access server 524 checks if the packet is a retrieve request for the user access profile. If the packet is a retrieve request, the access server 524 generates and transmits a request to retrieve the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then creates the user session ID and updates the user entries of the user access profile with the user ID and date/time updated in step 1312. The access server 524 then transmits a reply message with the session ID and the location of the LDAP server in step 1314 before terminating at step 1326.

If the packet is not a retrieve request, the access server 524 checks if the packet is a release request in step 1316. If the packet is a release request, the access server 524 retrieves the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then updates the user session ID and updates the user entries of the user access profile with the user ID and date/time updated in step 1318. The access server 524 then transmits a reply message with the session ID in step 1320 before terminating at step 1326.

If the packet is not a release request, the access server 524 checks if the packet is an update request in step 1322. If the packet is not an update request, the access server 524 registers an invalid profile request security event before terminating at step 1326. If the packet is an update request, the access server 524 retrieves the user access profile from the local database system 570 for local user access profiles or from the local database system 590 for foreign user access profiles. The access server 524 then updates the user access profile with the information in the packet in step 1324. The access server 524 then transmits a complete message to the user network 510 to signify the profile update is complete before terminating at step 1326. In one embodiment, the local database system 570 uses the user access profile system 572 to retrieve and store the local user access profiles and the local database system 590 uses the user access

16

profile system 592 to retrieve and store the foreign user access profiles.

User, Device, and Service Sessions

Access network providers provide user and device sessions in addition to service sessions to distinguish users when providing communication services. A user session is the information associated with a user accessing a network. A device session is the information associated with a device being used to access a network. A service session is the information associated with a service being provided over a network. Service providers distinguish users instead of access devices or links. This allows multiple users to share a single access device with each user receiving their own customized or preferred services. Advantageously, service providers establish service access rights and restrictions such as preventing adult content for younger viewers or sharing an access device for business and personal use. Also, user, device, and service sessions allow a service provider to group multiple service providers to provide a composite of services to the user similar to a contractor/sub-contractor relationship.

FIGS. 5, 14, 15, and 16 disclose one embodiment for device, user, and service sessions in an example of the invention. FIG. 14 depicts a logical view of device, user, and service sessions in an example of the invention. Devices 1402, 1404, and 1406 are comprised of session type specific information and session links. Session type specific information include public keys, private keys, and session ID. Session links include user sessions, device sessions, and service sessions. The public keys and private keys are for encryption and decryption of messages. Session ID identifies the session ID for the device. User sessions are user sessions that the device is logically linked to. Service sessions are the service sessions the device is logically linked to. Users 1408, 1410, and 1412 are comprised of public keys, private keys, session ID, device sessions, and service sessions. Session ID identifies the session ID for the user. Device sessions are the device sessions the user is logically linked to. Service sessions are the service sessions the user is logically linked to. Services 1414, 1416, and 1418 are comprised of public keys, private keys, session ID, user sessions, device sessions, and sub service sessions. Session ID identifies the session ID for the service. User sessions are the user sessions the service is logically linked to. Device sessions are the device sessions the service is logically linked to. Sub service sessions are the service sessions the service is logically linked to.

In one example, device 1402 is linked to user B 1410 via a link 1422. User B is also linked to service N 1418 via a link 1424. Device 1402 contains the user information in the user session fields for user B 1410 and the service information in the service session fields for service N. User B 1410 contains the device information in the device session fields for device 1402 and the service information in the service session fields for service N 1418. Service N 1418 contains the device information in the device session fields for device 1402 and the user information in the user session fields for user B 1410. Service N 1418 is also linked to service A 1414 via a link 1426. Service N 1418 contains additional service information in the service session fields for service A 1414. Service A 1414 contains the composite service information in the service session fields for service N 1418. Service A 1414 also includes an owning service which is a reference to the service that owns service A.

FIG. 15 depicts a flowchart for a user based session in an example of the invention. FIG. 15 begins in step 1500. In step 1502, the network device 512 establishes a connection

17

with the access server 524. The access server 524 generates and transmits a user ID query to the user network 510 in step 1504. In step 1506, the network device 512 receives the user ID query and transmits a user ID reply to the access server 524. The access server 524 receives and processes the user ID reply to generate a device session. The network device 512 then transmits a packet to the access server 524. The access server 524 then receives the packet in step 1508. In step 1510, the access server 524 processes the packet to check if the packet includes a user session ID.

If the packet does not include the user session ID, the access server 524 transmits a login query encrypted by the network device's 512 public encryption key 1402 to the network device 512 in step 1512. The network device 512 decodes the login query with its public encryption key and transmits a login reply encrypted with its public encryption key in step 1516. The access server 524 then decodes the login reply with the private encryption key and checks if the user ID is valid. If the user is valid, the access server 524 generates a user session and a session ID in step 1520. In some embodiments, the user session ID is the original IP address. The access server 524 then encrypts with the public encryption key and transmits a complete reply with the user session ID to the network device 512 in step 1522 before returning to step 1508.

If the packet does include the user session ID, the access server 524 checks if the user and user session ID are valid in step 1514. If the user or user session ID is not valid, the access server 524 discards the packet and registers a security event in step 1515 before returning to step 1508. If the user and user session ID are valid, the access server 524 generates a service request with the user session ID and the service session ID if available. The access server 524 encrypts the service request with the public encryption key where appropriate. The access server 524 transmits the packet including the service request in step 1518 before returning to step 1508.

FIG. 16 depicts a flowchart for a service based session in an example of the invention. FIG. 16 begins in step 1600. The network device 512 transmits a service request to the access server 524. The access server 524 then receives the service request in step 1602. In step 1604, the access server 524 checks if the service request includes a service session ID.

If the service request does not include the service session ID, the access server 524 transmits a service ID query encrypted with the public encryption key to the network device 512 in step 1606. The network device 512 decodes the service ID query with its private key and transmits a service reply encrypted with the service public encryption key to the access server 524 in step 1610. The access server 524 then decodes the service reply with the service private encryption key and checks if the user is valid. If the user is valid, the access server 524 generates a service session and a session ID in step 1614. The access server 524 then transmits a complete reply with the service session ID to the network device 512 in step 1616 before returning to step 1602.

If the packet does include the service session ID, the access server 524 checks if the user and service session ID are valid in step 1608. In one embodiment, the service session ID is the destination IP address. If the user or service session ID is not valid, the access server 524 discards the packet and registers a security event before returning to step 1602. If the user and service session ID is valid, the access server 524 updates the service request with the user session ID and the service session ID. The access server 524

18

encrypts the service request with the service public encryption key where appropriate. The access server 524 transmits the service request with the user session ID and the service session ID in step 1612 before returning to step 1602.

Access Network Security—FIGS. 5, 17, and 18 Service Capability Firewall

A network service provider interfaces with a network access provider at a transport functional level. The interface typically includes Internet protocol firewalls to provide security. Unfortunately, the interface between the network service provider and the network access provider is not able to hide the implementation details such as addressing schemes, transport details, and equipment specifics. The hiding of implementation details is preferred to prevent hackers from manipulating the implementation details. FIGS. 5, 17 and 18 depict one embodiment for a service capability firewall in an example of the invention. In this embodiment, the interface between the network service provider and the network access provider occurs at a named functional level instead of the transport addressing level. A named function request is any request for a capability of service provided by the network service provider. This allows the network service provider to hide the implementation details. The network service provider exposes only functional capabilities to the users depending on their security rights.

FIG. 17 depicts a flowchart for the firewall 556 for service capability firewall in an example of the invention. FIG. 17 begins in step 1700. In step 1702, the firewall 556 waits for information. In this embodiment, the information is in the form of a packet. The firewall 556 receives and processes a packet including a named function request from the network device 512 in step 1704. The firewall 556 then checks whether the firewall 556 is the destination for the named function request in step 1706. If the firewall 556 is not the destination, the firewall 556 registers a network request security event in step 1710. In step 1712, the firewall 556 encapsulates the packet with path information and transmits the packet to the database system 522 before returning to step 1702.

If the firewall 556 is the destination, the firewall 556 checks whether the sending address is consistent with the path in step 1714. If the sending address is not consistent with the path, the firewall 556 returns to step 1710. If the sending address is consistent with the path and does not belong to the accessing network, the firewall 556 checks if the sending address/session ID/named function combination is cached in step 1716. If the sending address/session ID/named function combination is cached, the firewall 556 replaces the packet's named function request with the private cached destination address in step 1718. The firewall 556 then checks if the private destination address is known and allowed to pass in step 1720. If the destination is known and allowed to pass, the firewall 556 transmits the packet on the correct path with standard firewall access in step 1722 before returning to step 3802. If the destination is not known or not allowed to pass, the firewall 556 returns to step 1710.

If the sending address/session ID/named function combination is not cached, the firewall 556 set an external request timer in step 1724. The firewall 556 then generates and transmits a request for the database system 522 in step 1726. The firewall 556 checks whether a reply is received or the timer has expired in step 1728. If the reply is not received and the timer has not expired, the firewall 556 returns to step 1728. In another embodiment, a blocked I/O is used instead of the wait loop in step 1728. If the reply is received or the timer has expired, the firewall 556 checks if there is a valid

19

reply in step 1730. If the reply is invalid or no reply was received, the firewall 556 discards the packet and registers a translation failure event in step 1734 before returning to step 1702. If the reply is valid, the firewall 556 replaces the packet's named function request based on the reply and caches the address/session functions returned in step 1732 before returning to step 1720.

FIG. 18 depicts a flowchart for the database system 522 for service capability firewall in an example of the invention. FIG. 18 begins in step 1800. In step 1802, the database system 522 waits for the next request. The database system 522 then receives and processes the request in step 1804. In step 1806, the database system 522 then checks if the access server requester is known. If the access server requester is not known, the database system 522 registers an unknown requester event in step 1808. The database system 522 then appends the reply with the packet and replies to the access server 524 in step 1810. The database system 522 then generates and transmits a decline reply to the access server 524 in step 1812 before returning to step 1802.

If the requester is known, the database system 522 then checks if the request is a capability appeal in step 1814. A capability appeal is any message, signaling or instruction requesting a capability with a related network address in the service provider. If the request is not a capability appeal, the database system 522 then checks if the request is a security event in step 1816. If the request is not a security event, the database system 522 register an unknown request event in step 1817 before returning to step 1810. If the request is a security event, the database system 522 increments a path/device security record and appends the requester information to the request packet and logs the event in step 1822 before returning to step 1802. If the request is a capability appeal, the database system 522 then identifies the user and network device 512 in step 1818. The database system 522 then checks if the access is valid in step 1822. If the access is invalid, the database system 522 returns to step 1822. If the access is valid, the database system 522 retrieves the user and network device profiles in step 1824. The database system 522 then checks whether the capability is valid for the user and session state in step 1826. If the capability is invalid, the database system 522 returns to step 1822. If the capability is valid, the database system 522 generates a session ID, the network address of the capability requested, and functions available and appends the network address, which is only sent to the firewall/router 526 and not to the user, to the reply in step 1830. The database system 522 then transmits the approve reply to the access server 524 in step 1832 before returning to step 1802.

Access Network Service Authorization—

FIGS. 5, 19, 20, 21, 22, 23, and 24

Prepaid Access

Prepaid phone cards are commonly used in PSTN, where the customer pays a prepaid amount that is debited against when the customer makes a call. In data networks, prepaid cards are not currently being used. FIGS. 5 and 19 disclose one embodiment for prepaid access in an example of the invention. In this embodiment, users buy prepaid cards from network providers. When the user requests access to one of many access providers throughout the country, the access provider verifies the prepaid account code before providing the access. A prepaid account code is any number that relates to a user's prepaid account. The prepaid account is debited against for the charges related to the access. Other charges related to the service provided may also be debited against the prepaid account. For example, a user may purchase an item from a website and have the charges debited against the

20

prepaid account. Once the prepaid amount is reached, the access provider terminates the access to the user. In one embodiment, the network provider provides different levels of service such as gold, silver, and bronze. The gold service has guaranteed throughput but higher rates for access, while the bronze service has lower throughput and rates.

FIG. 19 depicts a flowchart for prepaid access in an example of the invention. FIG. 19 begins in step 1900. In step 1902, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 1904. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits information identifying a billing code. A billing code is any number that identifies a user for billing. Two examples of billing codes are prepaid account codes and credit card numbers. In this embodiment, the information identifying a billing code is a response including a prepaid account code to the login query. In other embodiments, the billing code is retrieved from the user access profile. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response including the prepaid account code to determine if the user is known in the local database system 520 in step 1906. A user is known when the user is allowed to use the access network 520. In one embodiment, the user is known in the local database system 520 if there is time/amounts left in the user's prepaid account. In another embodiment, the database system 522 evaluates a positive balance file to determine the remaining time/amount in the user's prepaid account. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 1908 before returning to step 1902.

If the user is not known, the database system 522 checks if there is an appeal server for user authentication in step 1910. In one embodiment, the access server 524 performs the appeal on a decline. If there is no appeal server, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2412 before returning to step 2402. If there is an appeal server, the database system 522 generates an authorization query including the prepaid account code for the central database system 580 in step 1914. The database system 522 checks if the user is known in the central database system 580 in step 1914. In one embodiment, the user is known in the central database system 580 if there is time/amounts left in the user's prepaid account. If the user is known, the database system 522 proceeds to step 1908. If the user is not known, the database system 522 proceeds to step 1910.

In one embodiment, the database system 522 uses a user authorization system 575 for checking if the user is known in the local database system 520. The user authorization system 575 contains all the prepaid customers, prepaid customer information, prepaid account codes, and the amount/quantity remaining in the prepaid account to verify if access is allowed. In one embodiment, the database system 522 uses a user authorization system 581 as the appeal server for checking if the user is known in the local database system 580. The user authorization system 581 contains all the prepaid customers, prepaid customer information, and the amount/quantity remaining in the prepaid account.

Bank Card Access

In prior systems, access providers authenticated users using their own databases. FIGS. 5, 20 and 21 disclose one

embodiment for bank card access in an example of the invention. In this embodiment, access providers authenticate users through bank card financial networks using the users' credit card numbers. Network providers use credit or debit card numbers as user ID and passwords for authentication and authorization purposes. Users use prepaid cards, phone cards, and credit cards in PSTN. However, no bank cards have been used for access to data networks other than for batch bill payment.

FIG. 20 depicts a flowchart for bank card access for a connection in an example of the invention. FIG. 20 begins in step 2000. In step 2002, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2004. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits information identifying a billing code. In this embodiment, the information identifying a billing code is a response including credit card numbers to the login query. In other embodiments, the billing code is retrieved from the user access profile. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response to see if the user is known in the local database system 570 in step 2006. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 2008 before returning to step 2002.

If the user is not known, the database system 522 processes the response to check if the user is identified by credit card numbers in step 2010. If the user is not identified by the credit card numbers, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2012 before returning to step 2002. If the user is identified by the credit card numbers, the database system 522 generates an authorization query as a pre-authorization hold or authorization/capture transaction for the central database system 580 in step 2014. In one embodiment, the central database system 580 uses a financial interface 582 to interface with a financial switch to banks for authentication and authorization. The database system 522 then checks if the user is authenticated and authorized by the central database system 580 in step 2016. If the user is authenticated and authorized, the database system 522 logs access information and an authorizing financial entity or institution in step 2018 before proceeding to step 2008. If the user is not known, the database system 522 checks if there is another database system for authorization such as for foreign user access in step 2018. If there is another database system, the database system 522 returns to step 2014. If there is not another database system, the database system 522 returns to step 2012.

FIG. 21 depicts a flowchart for bank card access for a disconnection in an example of the invention. FIG. 21 begins in step 2100. In step 2102, the access server 524 waits for a disconnection. The network device 512 then disconnects from the access server 524 in step 2104. The access server 524 then checks whether the user was on a pre-authorization hold in step 2106. If the user is not on a pre-authorization hold, the 522 logs access information and an authorizing database system 522 in step 2108 before proceeding to step 2102. If the user is on a pre-authorization hold, the database system 522 generates a pre-authorization complete transaction for the central database system 580 in step 2110 before returning to step 2108.

Access Cards

In prior systems, access providers authenticated users using their own databases. FIG. 5, 22 and 23 disclose one embodiment for network access cards in an example of the invention. An access card is a card that a user uses to access a network. The access card includes an access card account code. The access card account code is any number that relates to the user's access account. In this embodiment, access providers authenticate users who have access cards using other access providers' databases. Access providers then may enter into sharing agreements with other access providers. These sharing agreements allow users to use multiple access providers, which provide the users greater mobility and availability. Users use phone cards in PSTN for phone card access. However, no access cards for data networks have been used.

FIG. 22 depicts a flowchart for network access cards for a connection in an example of the invention. FIG. 22 begins in step 2200. In step 2202, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2204. In another embodiment of the invention, the user calls a toll free number to request access. A service control point is queried to determine where to route the request for access similar to an automatic call distribution (ACD). The request for access is then routed to the access server 524 to establish a connection between the network device 512 and the access server 524.

Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits a response including the access card account code to the login query. The access server 524 then transfers the response from the network device 512 to the database system 522. In this embodiment, the database system 522 checks if the user is allowed to use the access network 520 by checking whether the user is known. A user is known when the user is allowed to use the access network 520. For example, a user is known when their user access profile is located within the local database system 570 and their user access profile allows the user to use the access network 520. The database system 522 receives and processes the response including the access card account code to check if the user is known in the local database system 570 in step 2206. If the user is known, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 2208 before returning to step 2202.

If the user is not known, the database system 522 checks if the response included foreign network account information in step 2210. Foreign network account information is any information that is indicative of an account that is external to local database system 570 that the user is attempting to gain access. If there is no foreign network account information, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2212 before returning to step 2202.

If there is foreign network account information, the database system 522 identifies the local database system 590 based on the foreign network account information and generates an authorization query for the local database system 590 in step 2214. The database system 522 then checks if the user is authenticated and authorized by the local database system 590 in step 2216. If the user is authenticated and authorized, the database system 522 logs contract and settlements information returned by the local database system 590 or indicated by the database system 522 in relation to local database system 590 in step 2218 before proceeding to step 2208. If the user is not known, the

23

database system 522 proceeds to step 2212. In one embodiment, the local database system 570 uses the user authorization system 575 to check if the user is known in the local database system 570. In one embodiment, the local database system 590 uses the user authentication system 593 for authentication and authorization.

FIG. 23 depicts a flowchart for network access cards for a disconnection in an example of the invention. FIG. 23 begins in step 2300. In step 2302, the access server 524 waits for a disconnection. The network device 512 then disconnects from the access server 524 in step 2304. The access server 524 then generates and transmits a logoff query for the database system 522. The database system 522 then transfers the logoff query to the local database system 570 and the local database system 590. The database system 522 logs the access information and the authorizing database system in step 2308 before returning to step 2302.

Global Authentication

In prior systems, access providers authenticated users using their own databases. FIGS. 5 and 23 disclose one embodiment for global authentication in an example of the invention. In this embodiment, access providers authenticate users using a centralized database. Access providers then may enter into sharing agreements with other access providers. These sharing agreements allow users to use multiple access providers, which provide the users greater mobility. Currently, cellular telephone companies enter into these sharing arrangements for greater coverage. However, this sharing of databases for authentication and authorization has not occurred for data networks.

FIG. 24 depicts a flowchart for global access in an example of the invention. FIG. 24 begins in step 2400. In step 2402, the access server 524 waits for a connection. The network device 512 connects to the access server 524 in step 2404. Once a connection is established, the database system 522 generates and transmits a login query for the network device 512. The network device 512 transmits a response to the login query. The access server 524 then transfers the response from the network device 512 to the database system 522. The database system 522 receives and processes the response to see if the user is native in the local database system 570 in step 2406. In this embodiment, the database system 522 checks if the user is allowed to use the access network 520 by checking whether the user is native. A user is native when the user is allowed to use the access network 520. For example, a user is native when their user access profile is located within the local database system 570 and their user access profile allows the user to use the access network 520. If the user is native, the database system 522 logs the access information and authorizes to provide access to the user via the access server 524 in step 4308 before returning to step 2402.

If the user is not native, the database system 522 checks if there is an authentication/authorization server in the database system 522 for a foreign network for user authentication in step 1910. If there is no authentication/authorization server in the database system 522 for the foreign network, the database system 522 transmits an instruction to the access server 524 to refuse the user login and to disconnect the network device 512 from the access server 524 in step 2412 before returning to step 2402. If there is an authentication/authorization server in the database system 522 for the foreign network, the database system 522 generates an authorization query for the central database system 580 in step 2414. The database system 522 then checks if the user is known in the central database system 580 in step 2414. If the user is known, the database

24

system 522 proceeds to step 2408. If the user is not known, the database system 522 proceeds to step 2410. In one embodiment, the central database system 580 uses a user authorization system 581 to check if the user is known.

Access Network Proxy/Environment—

FIGS. 5, 25, 26, 27, 28, 29, 30, and 31

User Based Proxy

A proxy is an application that represents itself as one or more network endpoints. The proxy receives a request for services from a user at a network endpoint and acts on behalf of the user in transmitting and receiving user requests and replies. There are Internet proxy agents that are user network access bind points. However, the Internet proxy agents are not user specific, and they act to protect user interests, not network. These proxy agents provide address translation and basic firewall functionality. Client focused proxies have extended to cookie collection and password handling.

FIGS. 5, 25 and 26 disclose one embodiment for user based proxies in an example of the invention. The user based proxies obtain information for the user and establishes a user specific network presence. A benefit of having a user specific network presence is that user access is handled by a process owned by a network security certificate authority that prevents Trojan horse network attacks. User based proxy provides a single control and monitor point for a user. The proxy agents provides a bind point for all user specific access such as user profile functionality, translations, security, and caching.

FIG. 25 depicts a flowchart for the access server 524 for user based proxies in an example of the invention. FIG. 25 begins in step 2500. In step 2502, the access server 524 waits for the next event. If the next event is a connection, the network device 512 connects with the access server 524 in step 2504. In step 2506, the access server 524 then collects user information and transfers the user information to the database system 522 to perform the security. In one embodiment, the database system 522 uses a security server 576 to perform the security. The access server 524 then checks if the user is allowed access in step 2508. If the user is not allowed access, the access server 524 disconnects the user in step 2510 before returning to step 2502.

If the user is allowed access, the access server 524 checks if a proxy is available in the database system 522 in step 2512. In one embodiment of the invention, the user proxies are in the host system 575. If the proxy is available, the access server 524 proceeds to step 2516. If the proxy is not available, the access server 524 starts the proxy in the database system 522 in step 2514 before proceeding to step 2516.

The access server 524 checks if the user access profile information is available in step 2516. If the user access profile information is not available, the access server 524 generates and transmits a user profile error to the network device 512 in step 2518 before returning to step 2502. If the user profile information is available, the access server 524 configures the proxy for the user in step 2520. The access server 524 then generates and transmits a message with the address of the user proxy and the public encryption key to the network device 512 in step 2522. The access server 524 also collects and transmits the access address of the network device 512 and the public encryption key to the user proxy. The access server 524 then returns to step 2502.

If the next event is a disconnection, the network device 512 disconnects with the access server in step 2524. The access server 524 then generates and transmits a reset command to the user proxy to clear the proxy state information and configuration in step 2526 before returning to

step 2502. If the next event is a status/request event from the user proxy, the access server 524 sets a user proxy reply timer in step 2528. The user proxy has written a status to the access server 524, and the user proxy receives a reply in step 2530 before returning to step 2502. If the next event is a timer expiration, the access server 524 receives the user proxy reply timer expiration in step 2532. The access server 524 then generates and transmits a continue wait message and status to the user proxy in step 2534 before returning to step 2528.

FIG. 26 depicts a flowchart for a user proxy for user based proxies in an example of the invention. FIG. 26 begins in step 2600. In step 2602, the user proxy waits for the next event. If the next event is the completion of the initialization, the user proxy checks if the initialization is complete in step 2604. The user proxy then sets a request timer in step 2606. The user proxy then request work from the access server 524 and transmits a status to the access server 524 in step 2608 before returning to step 2602. If the next event is the expiration of the request timer, the user proxy checks if the request timer expired in step 2610. The user proxy then registers a timeout in step 2612 before returning to step 2606.

If the next event is an access server 524 reply, the user proxy receives the access server 524 reply in step 2614. The user proxy then checks if the reply is a continue in step 2616. If the reply is a continue, the user proxy returns to step 2606. If the reply is not a continue, the user proxy processes the configuration information or action in step 2618 before returning to step 2606. If the next event is user request, the user proxy receives the user request for service in step 2620. The user proxy then checks if the user is valid in step 2622. If the user is valid, the user proxy then checks if the request is valid in step 2624. If the request is valid, the user proxy packages the request with the security certification and encryption in step 2626. The user proxy then transmits the request to the appropriate network destination in step 2628 before returning to step 2602. If the user or request is not valid, the user proxy registers a security event in step 2630 before returning to step 2602. The user proxy exchanges user information between the user network 510 and the appropriate network destination.

Subscriber Based Proxy

Another type of proxy is a subscriber based proxy. A subscriber is a logical entity such as an organization, corporation, or other grouping of users that has subscribed to services with a service provider. FIGS. 5, 27 and 28 disclose one embodiment for subscriber based proxies in an example of the invention. The subscriber based proxies obtain information for a user of a subscriber group and establishes a subscriber specific network presence. The benefit of having a subscriber specific network presence is that user access rights of the subscriber can be handled as a group, and group rights are owned by a network security certificate authority. The subscriber proxy agents provide a bind point for all subscriber specific access such as user profile functionality, translations, security, and caching.

FIG. 27 depicts a flowchart for the access server 524 for subscriber based proxies in an example of the invention. FIG. 27 begins in step 2700. In step 2702, the access server 524 waits for the next event. If the next event is a connection, the network device 512 connects with the access server 524 in step 2704. In step 2706, the access server 524 then collects user information and transfers the user information to the database system 522 to perform the security. In one embodiment, the database system 522 uses a security server 574 to perform the security. The access server 524

then checks if the user is allowed access in step 2708. If the user is not allowed access, the access server 524 disconnects the user in step 2710 before returning to step 2702. If the user is allowed access, the access server 524 checks if all required subscriber proxies are available in the database system 522 in step 2712. In one embodiment of the invention, the subscriber proxies are in the host system 575. If the proxies are available, the access server 524 proceeds to step 2716. If the proxies are not available, the access server 524 starts the required proxies in the database system 522 in step 2714 before proceeding to step 2716.

In step 2716, the access server 524 checks if the subscriber access profile information is available. If the subscriber access profile information is not available, the access server 524 generates and transmits a subscriber profile error to the network device 512 in step 2718 before returning to step 2702. If the subscriber profile information is available, the access server 524 configures the subscriber proxies for the subscriber in step 2720. The access server 524 then generates and transmits a message with the address of the subscriber proxy and the public encryption key to the network device 512 in step 2722. The access server 524 also collects and transmits the access address of the network device 512 and the public encryption key to the subscriber proxies. The access server 524 then returns to step 2702.

If the next event is a disconnection, the network device 512 disconnects with the access server in step 2724. The access server 524 then generates and transmits a reset command to the subscriber proxy in step 2726 before returning to step 2702. If the next event is a status/request event from the subscriber proxy, the access server 524 sets a subscriber proxy reply timer in step 2728. In step 2730, the subscriber proxy has written a status to the access server 524, and the subscriber proxy receives a reply before returning to step 2702. If the next event is a subscriber proxy timer expiration, the access server 524 receives the subscriber proxy reply timer expiration in step 2732. The access server 524 then generates and transmits a continue wait message and a status to the subscriber proxy in step 2734 before returning to step 2730.

FIG. 28 depicts a flowchart for a subscriber proxy for subscriber based proxies in an example of the invention. FIG. 28 begins in step 2800. In step 2802, the subscriber proxy waits for the next event. If the next event is the completion of the initialization, the subscriber proxy checks if the initialization is complete in step 2804. The subscriber proxy then sets a request timer in step 2806. The subscriber proxy then request work from the access server 524 and transmits a status to the access server 524 in step 2808 before returning to step 2802. If the next event is the expiration of the request timer, the subscriber proxy checks if the request timer expired in step 2810. The subscriber proxy then registers a timeout in step 2812 before returning to step 2806.

If the next event is an access server 524 reply, the subscriber proxy receives the access server 524 reply in step 2814. The subscriber proxy then checks if the reply is a continue in step 2816. If the reply is a continue, the subscriber proxy returns to step 2806. If the reply is not a continue, the subscriber proxy processes the configuration information or action in step 2818 before returning to step 2806. If the next event is a user request for service, the subscriber proxy receives the user request in step 2820. The subscriber proxy then checks if the user is valid in step 2822. If the user is valid, the subscriber proxy then checks if the request is valid in step 2824. If the request is valid, the subscriber proxy packages the request with the security

certification and encryption in step 2826. The subscriber proxy then transmits the request to the appropriate network destination in step 2828 before returning to step 2802. If the user or request is not valid, the subscriber proxy registers a security event in step 2830 before returning to step 2802. The subscriber proxy exchanges user information between the user network 510 and the appropriate network destination.

Dynamic Proxies

One prior system named "pluxy" allows a manual extension of a proxy. Pluxy requires manual determination and loading of a dynamic set of services, which is done on an operator basis. FIGS. 5 and 29 disclose one embodiment for dynamic proxies in an example of the invention. In this embodiment, both the user based proxies and the subscriber based proxies are extended in response to user actions. The dynamic proxy can be modified and enhanced to evolve with new services and/or protocols. The dynamic proxy size is smaller and more efficient by supporting only the logic that is being used. Implementation and development times of new service and protocol are also reduced because new proxies are not required with the new service and/or protocol.

FIG. 29 depicts a flowchart for a dynamic proxy for dynamic proxies in an example of the invention. FIG. 29 begins in step 2900. In step 2902, the dynamic proxy waits for the next event. If the next event is the completion of the initialization, the dynamic proxy checks if the initialization is complete in step 2904. The dynamic proxy then sets a request timer in step 2906. The dynamic proxy then request work from the access server 524 in step 2908 before returning to step 2902. If the next event is the expiration of the request timer, the dynamic proxy checks if the request timer expired in step 2910. The dynamic proxy then registers a timeout in step 2912 before returning to step 2906.

If the next event is an access server 524 reply, the dynamic proxy receives the access server 524 reply in step 2914. The dynamic proxy then checks if the reply is a continue in step 2916. If the reply is a continue, the dynamic proxy returns to step 2906. If the reply is not a continue, the dynamic proxy processes the configuration information in step 2918 before returning to step 2902. If the next event is a user request, the dynamic proxy receives the user request in step 2920. The dynamic proxy then checks if the user and the request are valid in step 2922. If the user and the request are valid, the dynamic proxy checks if there is a new service or protocol requested in step 2924. If there is a new service or protocol, the dynamic proxy proceeds to step 2934. If there is not a new service or protocol, the dynamic proxy packages the request with the security certification and encryption in step 2926. The dynamic proxy then transmits the request to the appropriate network destination in step 2928 before returning to step 2902. If the user or request is not valid, the subscriber proxy registers a security event in step 2930 before returning to step 2902.

If the next event is a new service/protocol request, the dynamic proxy receives the new service/protocol request in step 2932. In step 2934, the dynamic proxy processes the new service/protocol request to generate a handler request for a new service/protocol handler in step 2934 before returning to step 2926. The dynamic proxy receives the handler and executes the handler for the new service or proxy. In one embodiment, the dynamic proxy is enhanced by loading apple-like extensions from a handler system similar to a web browser.

Access Execution Environment

Proxy servers in the Internet represents a user in one network as existing in another network to obtain services for

the user. One problem with Internet proxy servers is the absence of the management of network resource utilization. In TINA-C, the User Agent and the User Session Manager are restricted to known service binding because any new service requires knowledge of or logic for a new CORBA interface. One problem with TINA-C is the absence of any provider resource management capabilities. Unfortunately, both Internet proxy servers and TINA-C do not provide any means of associating or viewing all network resources used by a user access session. Another problem is the lack of securing provider network resource against tampering by accessing users. Also, Internet proxy servers and TINA-C do not provide a platform-supported means of enforcing security levels on network access users. Both system also lack a validation of potential execution capability extensions.

FIGS. 5, 30, and 31 disclose one embodiment for an access execution environment in an example of the invention. The access execution environment easily and efficiently manages and secures network access by providing an execution environment in the access provider environment. Users access shared services and resources through their specific access execution environment. Because the access execution environment is in the access provider environment, the access provider can view the all of the user's activity by observing the execution environment. In this embodiment, the execution environment is setup as a standard system user on the network access platform, which allows management of the system user by operating system level user resource controls, quotas, and management mechanisms.

The access execution environment includes a multi-tasking virtual machine, a script interpreter, alias resolution capabilities, security certificate authentication, configuration handler, session caching, and protocol handling components. FIG. 30 depicts a block diagram of the access execution environment in an example of the invention. In FIG. 30, a network access platform 3000 comprises an execution environment manager 3010, a program and attribute database 3020, an execution environment A 3030, a transport A 3032, an execution environment B 3040, a transport B 3042, an execution environment C 3050, a transport C 3052, an execution environment D 3060, a transport D 3062, an execution environment E 3070, and a transport E 3072. In one embodiment, the network access platform 3000 is included within the database system 522. In another embodiment, the network access platform is included within the host system 575. The transport A 3032 is connected to the execution environment A 3030. The transport B 3042 is connected to the execution environment A 3040 and a workstation 3080. The transport C 3052 is connected to the execution environment C 3050. The transport D 3062 is connected to the execution environment D 3060. The transport E 3072 is connected to the execution environment E 3070. The execution environment A 3030 includes a transport handler 3034, a configuration handler 3036, and a security handler 3038. The execution environment B 3040 includes a transport handler 3044, a configuration handler 3046, and a security handler 3048. The execution environment C 3050 includes a transport handler 3054, a configuration handler 3056, and a security handler 3058. The execution environment D 3060 includes a transport handler 3064, a configuration handler 3066, and a security handler 3068. The execution environment E 3070 includes a transport handler 3074, a configuration handler 3076, and a security handler 3078.

In operation, a system administrator sets up a number of network access user accounts on the hardware platform. The administrator then sets up resource limits for each account/

platform resource. The administrator then starts an execution environment **3030**, **3040**, **3050**, **3060**, and **3070** for each access user account ID. The execution environment **A 3030** initializes and loads a configuration handler **3036**. The execution environment **A 3030** then loads the security handler **3038** and the transport handler **3034**. The transport handler **3034** opens a logical or physical transport port **A 3032** on the hardware platform using port information from the configuration handler.

FIG. 31 depicts a flow chart for the access execution environment in an example of the invention. FIG. 31 begins in step **3100**. In step **3102**, the security handler **3038** waits for a connect message from the transport handler **3034**. The security handler **3038** then receives the connect message from the transport handler **3034** in step **3104**. The security handler **3038** then generates and transmits a logon request via the transport handler **3034** in step **3106**. The security handler **3038** then checks if a reply for the logon request from the transport handler **3034** has been received in step **3108**. If the reply for the logon request has not been received, the security handler **3038** checks if a reply timeout has occurred in step **3110**. If a reply timeout has not occurred, the security handler **3038** returns to step **3108**. If a reply timeout has occurred, the security handler **3038** generates and transmits a disconnect message to the transport handler **3034** in step **3112** and returns to step **3102** to wait for a connect message.

In step **3114**, the security handler **3038** receives the reply for the logon request. Then the security handler **3038** retrieves the location of a security server from the configuration handler **3036** and transmits the logon information to the security server in step **3116**. The security handler **3038** then checks if a reply to the logon information from the security server has been received in step **3118**. If no reply has been received, the security handler **3038** checks if a reply timeout has occurred in step **3120**. If a reply timeout has not occurred, the security handler **3038** returns to step **3118**. If a reply timeout has occurred, the security handler **3038** checks if the logon information has been sent three times in step **3122**. The number of tries could be configurable. If the logon information has not been sent three times, the security handler **3038** returns to step **3116** to transmit the logon information. If the logon information has been sent three times, the security handler **3038** returns to step **3112** to send a disconnect message.

If the security server replied before the reply timeout, the security handler **3038** checks if the reply is an accept message in step **3124**. If the reply is a decline message, the security handler **3038** returns to step **3112**. If the reply is an accept message, the security handler **3038** transmits the accept message configuration parameters to the configuration handler **3036** in step **3126**. The configuration handler **3036** then loads attributes and programs and executes programs specified by the security server reply in step **3128**. The execution environment **A 3030** performs the execution of programs for the user in step **3130**. In another embodiment, the programs request attributes and/or programs to be loaded and/or executed. The programs include the ability to interface with users via the transport handler and load/execute other programs required by request types.

The configuration handler **3036** then checks if the transport handler **3034** receives a disconnect message in step **3132**. If the transport handler **3034** has not received a disconnect message, the configuration handler **3036** returns to step **3130**. If the transport handler **3034** has received a disconnect message, the transport handler **3034** transmits the disconnect message to the configuration handler **3036** in step

3134. In step **3136**, the configuration handler **6536** based on port configuration attributes closes the transport port **3032**, resets the transport handler **3034**, and notifies the execution environment manager **3010** to create a new execution environment for that port. The configuration handler **3036** gracefully shuts down all handlers, programs, and eventually the execution environment **A 3030**. The configuration handler **3036** transmits a shutdown message to the execution environment manager **3010** in step **3138** before terminating in step **3140**. The execution environment manager **3010** restarts the execution environment **A 3030** upon notification of the shutdown. The operations of the other execution environments **3040**, **3050**, **3060**, and **3070** perform in the same manner as the execution environment **A 3030** and are not discussed for the sake of clarity.

Access Network Translations—FIGS. 5, 32, 33, 34, 35, 36, 37, 38, and 39

Domain Name Scoping

The typing in of domain names such as www.nypostonline.com is quite cumbersome for the user to access specific services. The Domain Name Server (DNS) translates the domain name to a network address for the service. Clicking on bookmarks or favorites in browsers, which reference the domain names, does eliminate the need to type in the domain names. However, when users change network devices, these bookmarks or favorites do not follow the user. In an operating system environment, the command interpreter shell allows the user to customize interpretation of command strings. If no customization is loaded, the command interpreters interprets in a default fashion.

FIGS. 5, 32, and 33 disclose one embodiment for domain name scoping in an example of the invention. This embodiment provides a customizable network shell to overlay the standard DNS service offered in Internet Protocol based networks. Users then are able to define user specific acronyms or aliases for specific or logical services. An acronym or alias indicates domain names, macros, programs, actions, or network addresses. If the translation for the alias does not exist in the list of aliases for the user, the access server **524** checks if the alias exists in the list for a group in which the user belongs. The access server **524** then checks if the alias exist in the DNS for the general network. In another embodiment, the database system **522** checks the existence of the alias in the list for the group and in the DNS for the general network. There are numerous variations in the hierarchy for searching for an alias but are not discussed for the sake of simplicity.

FIG. 32 depicts a flowchart for the access server **524** for domain name scoping in an example of the invention. FIG. 32 begins in step **3200**. In step **3202**, the access server **524** waits for a packet. In this embodiment, the access server **524** receives information in the form of packets from the network device **512**. The access server **524** then receives and processes the packet from the network device **512** in step **3204**. The access server **524** then checks if the packet is an alias translation request in step **3206**. If the packet is not an alias translation request, the access server **524** transmits the packet with standard access logic in step **3214** before returning to step **3202**.

If the packet is an alias translation request, the access server **524** then checks if the alias translation is cached for the user in step **3208**. If the alias translation is cached, the access server **524** reformats the packet using the cached alias translation in step **3216** before returning to step **3214**. If the alias translation is not cached, the access server **524** sets an external request timer in step **3210**. The access server **524** then generates and transmits an alias translation request to

the database system 522 in step 3212. The alias translation request is any message, instruction, or signaling indicative of requesting an alias translation. The access server 524 checks whether a reply has been received or the external request timer has expired in step 3218. If a reply has not been received and the external request timer has not expired, the access server 524 returns to step 3218.

If a reply has been received or the external request timer has expired, the access server 524 checks if the reply was valid in step 3220. If the reply is invalid, the access server 524 discards the packet and replies with an invalid alias message or alias error message to the network device 512 before returning to step 3202. If the reply is valid, the access server 524 caches the alias translation for the user in step 3222 before returning to step 3216.

FIG. 33 depicts a flowchart for the database system 522 for domain name scoping in an example of the invention. FIG. 33 begins in step 3300. In step 3302, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 524 in step 3304. In step 3306, the database system 522 then checks if the access server requester is known in step 3306. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 3308. The database system 522 appends the reply information to the packet in step 3310. The database system 522 replies with a decline message to the access server 524 in step 3312 before returning to step 3302.

If the access server requester is known, the database system 522 then checks whether the request is an alias translation request in step 3314. If the request is not an alias translation request, the database system 522 registers an unknown request event in step 3316 before returning to step 3310. If the request is an alias translation request, the database system 522 identifies and authenticates the user and the network device 512 in step 3318. If the access is invalid, the database system 522 increments a path/device security record in step 3322. The database system 522 then appends the requester denial information to the request packet in step 3324. The database system 522 then replies with a decline message to the access server 524 in step 3312 before returning to step 3302.

If the access is valid, the database system 522 retrieves the user's alias translation list in step 3326. The database system 522 then translates the alias for the user and appends the translation to a reply to the access server 524. The database system 522 then replies with an approve message to the access server 524 in step 3330. In one embodiment of the invention, the database system 522 uses the alias translation system 578 to perform the operations disclosed in FIG. 33 and stores the aliases and alias translations in the alias translation system 578.

Inband Domain Name Service Lookup

The Domain Name Server (DNS) translations of a domain name to a network address delays user interaction with the service provider. In prior solutions in voice telephone Signaling System #7 networks, a signal transfer point performs an inband local number portability (LNP) lookup on a ISDN part call setup request such as an Initial Address Message (IAM). This inband lookup eliminates the network switch from launching a Transaction Capabilities Application Part (TCAP) LNP query to translate the telephone number in the IAM.

FIGS. 5 and 34 disclose one embodiment for inband domain name service in an example of the invention. This embodiment provides a cache for the access server 524 for alias translations on a per user basis. Thus, the external

queries for translations of aliases to domain names, macros, programs, or network addresses are eliminated. If the translated value of a request is in the alias translation cache, no translation request will be required to a DNS server. Therefore, users experience reduced delays when requesting a service.

FIG. 34 depicts a flowchart for the access server 524 for an inband domain name service lookup in an example of the invention. FIG. 34 begins in step 3400. In step 3402, the access server 524 waits for a packet. The access server 524 then receives and processes the packet from the network device 512 in step 3404. The access server 524 then checks if the destination network address and user is valid in step 3406. If the destination network address and user is valid, the access server 524 transmits the packet on the correct path to reach the destination in step 3408 before returning to step 3402.

If the destination network address or the user is invalid, the access server 524 then checks if the alias translation is cached for the user in step 3410. If the alias translation is cached, the access server 524 reformats the packet using the cached alias translation in step 3412 before returning to step 3402. If the alias translation is not cached, the access server 524 sets an external request timer in step 3414. The access server 524 then generates and transmits an alias translation request to the database system 522 in step 3416. The access server 524 checks whether a reply has been received or the external request timer has expired in step 3418. If a reply has been received and the external request timer has not expired, the access server 524 returns to step 3418.

If a reply has been received or the external request timer has expired, the access server 524 checks if the reply was valid in step 3420. If the reply is invalid, the access server 524 discards the packet and replies with an invalid alias message to the network device 512 in step 3424 before returning to step 3402. If the reply is valid, the access server 524 caches the alias translation for the user in step 3422 before returning to step 3412.

Inline Access Service Triggering

Typical firewalls filter packets out on a request by request basis on "what is not allowed". FIGS. 5 and 35 disclose one embodiment for inline access service triggering. In this embodiment, the access server 554 filters packets out on a "what is allowed" basis. Also, the access server 554 performs packet filtering on a service access basis.

FIG. 35 depicts a flowchart for inline access service triggering in an example of the invention. FIG. 35 begins in step 3500. In step 3502, the access server 554 waits for the next packet. In this embodiment, the access server 554 receives information in the form of packets from the network device 512. The access server 554 receives and processes a packet from the network device 512 in step 3504. In step 3506, the access server 554 checks if the protocol, sending address, and the destination address are known and allowed to pass. In one embodiment, the access server 554 checks the database system 522 for the determination made in step 3506. If the protocol, sending address, and the destination address are known and allowed to pass, the access server 554 checks if the sending address is consistent with the path in step 3508. If the sending address is not consistent with the path, the access server 554 registers a path security event in step 3510. The access server 554 then formats the packet with path information in step 3512. The access server 554 also increments the path/device security record and logs the event before returning to step 3502. If the sending address is consistent with the path, the access server 554 transmits the packet on the correct path with standard firewall access in step 3514 before returning to step 3502.

If the protocol, sending address, or the destination address are not known or not allowed to pass, the access server 554 checks if the request is a filter appeal in step 3515. If the request is not a filter appeal, the access server 554 discards the packet in step 3526 before returning to step 3502. If the request is a filter appeal, the access server 554 identifies and authenticates the user and the network device 512 in step 3516. The access server 554 then checks if the access is valid in step 3518. If the access is invalid, the access server 554 returns to step 3510. If the access is valid, the access server 554 retrieves the user access profile and network device's 512 profile in step 3520. The access server 554 then modifies the access logic for filter modification in step 3524. The access server 554 then modifies the protocol and address filters based on the request in step 4824 before returning to step 3514. In some embodiments, the access server 554 modifies the filters in conformance with the user access profile.

Access Service Triggering

Access providers sometimes need to extend their authentication logic beyond their primary access devices. No prior system in data networks extends the authentication logic beyond what is performed by the access provider's access devices. FIGS. 5, 36 and 37 disclose one embodiment for access service triggering. In this embodiment, the access server 524 triggers a request to external access control logic. Access providers extend the access and authentication logic beyond their access devices while maintaining centralized control of the authentication logic and user access profiles. Also, the access server 554 performs packet filtering on a service access basis.

FIG. 36 depicts a flowchart for the access server 524 for access service triggering in an example of the invention. FIG. 36 begins in step 3600. In step 3602, the access server 524 waits for the next packet. The access server 524 then receives and processes a packet from the network device 512 in step 3604. In step 3606, the access server 524 checks if the protocol, sending address, and the destination address are known and allowed to pass. In one embodiment, the access server 524 checks the database system 522 for the determination made in step 3606. If the protocol, sending address, and the destination address are known and allowed to pass, the access server 524 checks if the sending address is consistent with the path in step 3608. If the sending address is not consistent with the path, the access server 524 registers a path security event in step 3610. The access server 524 then formats the packet with path information in step 3612 before returning to step 3602. If the sending address is consistent with the path, the access server 524 transmits the packet on the correct path with standard firewall access in step 3614 before returning to step 3602.

If the protocol, sending address, or the destination address are not known or are not allowed to pass, the access server 524 sets an external request timer in step 3616. The access server 524 then generates and transmits a request with the path information to the database system 522 in step 3618. The access server 524 then checks if a reply has been received or the external request timer has expired in step 3620. If the reply has not been received or the external request timer has not expired, the access server 524 checks if the reply is valid in step 3622. If the reply is invalid, the access server 524 discards the packet in step 3624 before returning to step 3602. If the reply is valid, the access server 524 then modifies the protocol and address filters based on the reply in step 3624 before returning to step 3614.

FIG. 37 depicts a flowchart for the database system 522 for access service triggering in an example of the invention.

FIG. 37 begins in step 3700. In step 3702, the database system 522 waits for the next request. The database system 522 then receives and processes the request in step 3704. In step 3706, the database system 522 then checks if the access server requester is known in step 3706. If the access server requester is not known, the database system 522 registers an unknown requester event in step 3708. The database system 522 then appends the reply to the packet in step 3710. The database system 522 then generates and transmits a decline reply to the access server 524 in step 3712 before returning to step 3702.

If the access server requester is known, the database system 522 then checks if the request is a filter appeal in step 3714. If the request is not a filter appeal, the database system 522 then checks if the request is a security event in step 3716. If the request is not a security event, the database system 522 registers an unknown action event in step 3717 before returning to step 3712. If the request is a security event, the database system 522 increments a path/device security record, appends the requester information to the request packet, and logs the event in step 3718. The database system 522 then checks if the request is a security event in step 3719. If the request is not a security event, the database system 522 then returns to step 3712. If the request is a security event, the database system 522 proceeds to step 3728.

If the request is a filter appeal, the database system 522 then identifies the user and network device 512 in step 3720. The database system 522 then checks if the access is valid in step 3722. If the access is invalid, the database system 522 returns to step 3718. If the access is valid, the database system 522 retrieves the user and network device profiles in step 3724. The database system 522 then generates access logic and appends the access logic to a reply in step 3726. The database system 522 then transmits the approve reply to the access server 524 in step 3728 before returning to step 3702.

Personal URL

The Internet currently uses Uniform Resource Locator (URL) addresses such as www.yahoo.com so that the address is in more human readable form than the network address. Unfortunately, user cannot distinguish themselves by a network address because the user's network address changes when the user's access point changes. FIGS. 5, 38 and 39 disclose one embodiment for a personal URL. In this embodiment, a network user may publish their location on the network by a user alias. A user alias is any alias that relates to the network address of a user. Logically linking a user's current network address with a user alias allows a user to be located wherever the user accesses the network.

FIG. 38 depicts a flowchart for a personal URL lookup in an example of the invention. FIG. 38 begins in step 3800. In step 3802, the database system 522 waits for the next packet. The database system 522 then receives and processes the packet including the user alias. The database system 522 then checks if a user alias translation is cached for this user in step 3806. If the user alias translation is cached, the database system 522 replies with the current network address of the user in step 3808 before returning to step 3802.

If the user alias translation is not cached, the database system 522 sets an external request timer in step 3810. The database system 522 then generates and transmits a request for user alias translation logic in step 3812. In one embodiment, the database system 522 transmits the request for user alias translation logic to the database system that contains the user access profile. The database system 522

then checks whether a reply was received or the external request timer expired in step 3814. If the reply was not received and the external request timer did not expire, the database system 522 returns to step 3814. If a reply was received or the external request timer did expire, the database system 522 checks if the reply was valid in step 3816. If the reply was invalid, the database system 522 replies with an invalid name message in step 3824 before returning to step 3802.

If the reply was valid, the database system 522 checks if there is a current network address for the user in step 3818. If there is no current network address for the user, the database system 522 replies with user alias not currently in network message in step 3820 before returning to step 3802. If there is a current network address for the user, the database system 522 caches the user alias translation in step 3822 before returning to step 3808.

FIG. 39 depicts a flowchart for a personal URL update in an example of the invention. FIG. 39 begins in step 3900. In step 3902, the database system 522 wait for a request. The database system 522 then receives and processes the request to update the user alias in step 3904. In one embodiment, the database system receives the request from a database system that contains the user access profile. In another embodiment, the request comes from the access server 524. The database system 522 then checks if the requester is known in step 3906. If the requester is not known, the database system 522 registers an unknown requester event in step 3910. The database system 522 then transmits a reply with decline in step 3912 before returning to step 3902.

If the requester is known, the database system 522 then checks if the requester is allowed to update the user alias in step 3914. If the requester is not allowed, the database system 522 registers an unauthorized requester event in step 3916 before returning to step 3912. If the requester is allowed to update, the database system 522 then identifies and authenticates the user and current network address in step 3918. The database system 522 then checks if the access is valid in step 3920. If the access is invalid, the database system 522 increments the path/device security record in step 3922. The database system 522 also appends the requester information to the request and logs the event before returning to step 3912.

If the access is valid, the database system 522 then updates the user alias translation with the current network address in step 3924. The database system 522 then replies with an approve in step 3926 before returning to step 3902. In one embodiment of the invention, the database system 522 uses the personal DNS system 579 to perform the operations disclosed in FIGS. 38 and 39.

Access Network Caching—FIGS. 5, 40, 41, 42, 43, and 44

Predictive Caching

Users that are dialed into the network at a rate of 56 kbps or greater typically retrieve information at a rate of 2–4 kbps. Various equipment between the network access provider and the service provider cause this lower rate of transmission. Some examples of the equipment are network access provider equipment, service provider equipment, network backbone overloading, traffic shaping device, firewalls, and routers. One solution for compensating for the lower rate of transmission is caching. Computers typically contain a memory cache for specific application or devices. Firewalls and routers may also contain caches for data. Unfortunately, none of these caches are user specific.

FIGS. 5, 40, 41 and 42 disclose one embodiment of the invention for predictive end user caching. Predictive end

user caching advantageously improves user noticeable delays and slowdowns due to the lower transmission rate. The extension of data requests from the user network 510 beyond the access server 524 is eliminated when the data requested is cached in the database system. Also, using a predictive algorithm reduces delay by improving caching efficiency based on a user's predictable pattern.

FIG. 40 depicts a flowchart for the access server 524 for auditing in an example of the invention. FIG. 40 begins at step 4000. In step 4002, the user network 510 establishes a connection to the access server 524. In step 4004, the access server 524 generates and transmits a login request message to the database system 522. The access server 524 then checks if the database system allowed an access session to be established for the user in step 4006.

If the access session is not established, the access server 524 terminates in step 4020. If the access session is established, the access server 524 then checks if an access session tear down is requested in step 4012. If the access session tear down is requested, the access server 524 generates and transmits a tear down message with user and path information to the database system 522 in step 4014 before terminating in step 4020.

While no session tear down request is received, the access server 524 exchanges packets with the user network 510 depending on the service provided in step 4016. The access server 524 transmits all new packet destinations including the user and path information to the database system 522 in step 4018 before returning to step 4012. In one embodiment, the database system 522 stores the connect request message, the session reject message, the session accept message, the tear down message including the user and path information, and the packets including the user and path information in an audit database system 526. In one embodiment, the connect request message, the session reject message, the session accept message, the tear down message including the user and path information, and the packets including the user and path information are in the form of an audit message. An audit message is any message, information, or signaling that is audited while a user is accessing an access network 520.

FIG. 41 depicts a flowchart for the database system 522 for auditing in user predictive caching in an example of the invention. FIG. 41 begins in step 4100. In step 4102, the database system 522 waits for an audit message. In step 4104, the database system 522 receives and processes the audit message. The database system 522 then checks if the audit message is a session accept message or a tear down message in step 4106. If the audit message is not a session accept message or a tear down message, the database system 522 retrieves the session state object and updates the object with the audit message in step 4112 before returning to step 4102.

If the audit message is a session accept message or a tear down message, the database system 522 then checks if the audit message is a session accept message in step 4108. If the audit message is a session accept message, the database system 522 generates a session state object including the user, device, path, and session ID in step 4114 before returning to step 1802. If the audit message is a tear down message, the database system 522 retrieves the session state object and updates the object in step 4112. Also, the database system 522 stores the event in step 4116. In one embodiment, the database system 522 stores the event in the cache database system 574. The database system 522 removes the active reference from the session state object before returning to step 4102. In one embodiment of the invention, the database system 522 uses the audit database system 573 to perform the operations disclosed in FIG. 41.

FIG. 42 depicts a flowchart for the database system 522 for caching in user predictive caching in an example of the invention. FIG. 42 begins in step 4200. In step 4202, the database system 522 waits for the next event. The database system 522 processes the event in step 4204. In this embodiment, a request for data is a cache request. The database system 522 then checks if the event is a cache request in step 4206.

If the event is a cache request, the database system 522 identifies the user and retrieves the cached user data in step 4208. The database system 522 then checks if the requested data from the cache request is in the cached user data in step 4212. If the requested data is in the cached user data, the database system 522 replies with the requested data in step 4210 before returning to step 4202. If the requested data is not in the cached user data, the database system 522 retrieves the requested data from the service provider via the access server 524 in step 4214. The database system 524 then replies to the user network 510 with the requested data. Alternatively, in another embodiment, the access server 524 transfers the requested data from the service provider to the user network 510 and the database system 522.

In step 4222, the database system 522 updates the user access profile and the caching algorithm. The database system 522 then determines whether there is a change in the cached data based on the caching algorithm in step 4230. If there is no change in the cached data, the database system 522 returns to step 4202. If there is a change in the cached data, the database system 522 sets any new caching timers for the cached data in step 4236. The database system 522 also generates and transmits a translation change to the access server 524 to change the requested destination address to the database system 522 in step 4238 before returning to step 4202.

If the event is not a cache request, the database system 522 then checks if the event is an audit or timer request in step 4216. If the event is an audit or timer request, the database system 522 identifies the user and retrieves the user access profile in step 4218. The database system 522 then checks if the event is an audit request in step 4220. If the event is an audit request, the database system 522 proceeds to step 4222. If the event is not an audit request, the database system 522 refreshes the cached data from the service provider of the cached data based on the caching algorithm in step 4228. The database system 522 then resets the caching timer for the cached data in step 4234 before returning to step 4202.

If the event is not an audit or timer request, the database system 522 checks if the event is a reset event in step 4224. If the event is not a reset event, the database system 522 registers a unknown event received in step 4232 before returning to step 4202. If the event is a reset event, the database system 522 clears the user access profile statistics, caching timers, the user's cached data. In one embodiment, the user's cached data is stored in the cached database system 574 and the database system 522 uses the cached database system 574 to perform the operations disclosed in FIG. 42.

User Controlled Caching

Another solution to reduce user noticeable delays and slowdowns due to the lower transmission rate is user controlled caching. FIGS. 5, 40, 43 and 44 disclose one embodiment of the invention for user controlled caching. A user controls caching by creating a script of commands to cache data prior to the user accessing the network. For example, a user selects financial or local weather forecasts to cache, so when the user logs on to the network the information will be immediately available from the cache. Scripting has typi-

cally been done on general purpose computers. In a general purpose computer, the user sets up a sequenced set of commands to be executed when the script is executed. Unfortunately, this type of scripting has not been performed on a network cache.

In this embodiment of user controlled caching, the operation of the access server 524 is as described in FIG. 40. FIG. 43 depicts a flowchart for the database system 522 for auditing in user controlled caching in an example of the invention. FIG. 43 begins in step 4300. In step 4302, the database system 522 waits for an audit message. In step 4302, the database system 522 receives and processes the audit message. The database system 522 then checks if the audit message is a session accept message or a tear down message in step 4306. If the audit message is not a session accept message or a tear down message, the database system 522 retrieves the session state object and updates the object with the audit message in the cache in step 4312 before returning to step 4302.

If the audit message is a session accept message or a tear down message, the database system 522 then checks if the audit message is a session accept message in step 4308. If the audit message is a session accept message, the database system 522 generates a session state object in the cache including the user, device, path, and session ID in step 4314. The database system 522 then checks if the user has a pre-cache script in step 4318. If the user does not have a pre-cache script, the database system 522 returns to step 4302. If the user has a pre-cache script, the database system 522 generates and transmits a pre-cache instruction set to the access server 524 to change the translation filter to access the database system 522 for destinations in the pre-cache script so that any requests for those destinations are fulfilled from the cache in step 4320. The database system 522 then sets cache refresh timers in step 4322 and returns to step 4302.

If the audit message is a tear down message, the database system 522 retrieves the session state object and updates the object in step 4316. Also, the database system 522 stores the event in step 4316. In one embodiment, the database system 522 stores the session state object and the state in the cache database system 527. The database system 522 removes the active reference from the session state object before returning to step 4302. In one embodiment of the invention, the database system 522 uses the audit database system 573 to perform the operations disclosed in FIG. 43.

FIG. 44 depicts a flowchart for the database system 522 for caching in user controlled caching in an example of the invention. FIG. 44 begins in step 4400. In step 4402, the database system 522 waits for the next event. The database system 522 processes the event in step 4404. In this embodiment, a request for data is a cache request. The database system 522 then checks if the event is a cache request in step 4406.

If the event is a cache request, the database system 522 identifies the user and retrieves the cached user data in step 4408. The database system 522 then checks if the requested data from the cache request is in the cached user data in step 4412. If the requested data is in the cached user data, the database system 522 replies with the requested data in step 4410 before returning to step 4402. If the requested data is not in the cached user data, the database system 522 retrieves the requested data from the service provider via the access server 524 in step 4414. The database system 522 then replies to the user network 510 with the requested data. Alternatively, in another embodiment, the access server 524 transfers the requested data from the service provider to the user network 510 and the database system 522.

In step 4422, the database system 522 updates the user access profile and the caching algorithm. The database system 522 then determines whether there is a change in the cached data based on the caching algorithm in step 4430. If there is no change in the cached data, the database system 522 returns to step 4402. If there is a change in the cached data, the database system 522 sets the new caching timer for the cached data in step 4436. The database system 522 also generates and transmits a translation change to the access server 524 to change the requested destination address to the database system 522 in step 4440 before returning to step 4402.

If the event is not a cache request, the database system 522 then checks if the event is an audit or timer request in step 4416. If the event is an audit or timer request, the database system 522 identifies the user and retrieves the user access profile in step 4418. The database system 522 then checks if the event is an audit request in step 4420. If the event is an audit request, the database system 522 proceeds to step 4422.

If the event is not an audit request, the database system 522 checks if the event is a script timer in step 4428. If the event is not a script timer, the database system 522 refreshes the cached data from the service provider of the cached data based on the caching algorithm in step 4438 before returning to step 4402. If the event is a script timer, the database system 522 executes the user commands or command set specified by the script timer event in step 4434. The database system 522 then resets the script timer for the cached data in step 4442 before returning to step 4402.

If the event is not an audit or timer request, the database system 522 checks if the event is a reset event in step 4424. If the event is not a reset event, the database system 522 registers a unknown event received in step 4432 before returning to step 4402. If the event is a reset event, the database system 522 clears the user access profile statistics, caching timers, the user's cached data. In one embodiment, the user's cached data is stored in the cached database system 574 and the database system 522 uses the cached database system 574 to perform the operations disclosed in FIG. 44.

Access Network Switching— FIGS. 5, 45, 46, and 47

Switching Access by a User

Access between users and service providers vary based on quality and security, which in turn determine the costs of the access. Users typically have a need to switch between types of access depending on the service offered or what stage in the service the user is in. For example, a user browses the amazon.com website for books using a standard Internet access. When purchasing the books, the user needs a more secure Internet access to ensure that the user's credit card number is not stolen by a hacker.

One prior solution for enhanced security is data Virtual Private Networks (VPN). Data VPN's are relatively static constructions which allow the extension of a user network to another location by extending the network over leased lines or shared Internet/Intranet facilities. However, VPN's require service anticipation, planning and expense beyond what the typical Internet and Intranet users possess. VPN's also do not provide dynamic switching between accesses. Another prior solution is Local Area Network emulation (LANE). LANE utilizes ATM transports to extend the reach of the user network. However, most Internet and Intranet users do not possess ATM equipment and expertise, and the backbone network is still IP.

FIGS. 5 and 45 disclose one embodiment for switching access by a user in an example of the invention. A user

selects a different access to a service provider and is switched to the access without the user's connection to their access provider being interrupted. FIG. 45 depicts a flowchart for switching access by a user in an example of the invention. In this embodiment, a user using standard Internet access selects a premiere Internet access during the setup of a service application. There are numerous variations in switching between access paths such as from premiere Internet access path to a lower quality Internet access path, but only one embodiment is shown for the sake of simplicity.

FIG. 45 begins in step 4500. In step 4502, a user through the network device 512 transmits a request using a standard Internet access path through the service network 530 for a service application residing in the network device 562. The service application in the network device 562 receives the request and transmits a query for the quality of service (QOS) Internet access desired by the requester to the network device 512 in step 4504. The service application in the network device 562 then receives and processes a request to switch access to check if the user selected a premium QOS Internet access in step 4516. If the user did not select a premium QOS Internet access, the user and service application exchange packets using the standard Internet access via the network device 512, the access server 524, the service network 530, the access server 554, and the network device 562 in step 4516. The service application in the network device 562 returns the control to the user in step 4514 to select another service application in step 4502.

If the user selects a premium QOS Internet access, the service application in the network device 562 generates and transmits an authentication and authorization instruction for the premium QOS Internet access to the database system 522 to check if the switch of access is allowed. The database system 522 receives and processes the authentication and authorization instruction. The database system 522 then generates and transmits a premium access instruction to establish premium Internet access between the access server 524 and the access server 554 via the service network 540. In one embodiment, the database system 522 transmits the premium access instruction to the service network 540 to establish premium Internet access between the access server 524 and the access server 554. The database system 522 then generates and transmits the premium access instruction to the access server 524 to connect the network device 512 to the service network 540 for the premium Internet access. The database system 522 also generates and transmits the premium access instruction to the access server 554 to connect the network device 562 to the service network 540 for the premium Internet access.

Once the premium Internet access is established, the user and service application exchange packets using the premium Internet access via the network device 512, the access server 524, the service network 540, the access server 554, and the network device 562 in step 4512. The database system 522 then generates and transmits a disconnect instruction to the access server 524, the service network 540, and the access server 554 to tear down the premium Internet access in step 4518. The database system 522 then generates and transmits usage and performance statistics for the service application in the network device 562 in step 4520 before returning to step 4514. In one embodiment, the database system 522 uses a cross connect system 583 to perform the operations disclosed in FIG. 45.

Switching Access by a Service Provider

FIGS. 5 and 46 disclose one embodiment for switching access by a service provider in an example of the invention. The service provider selects a different access to a user and

41

is switched to the access without the user or service provider's connection to their access provider being interrupted. FIG. 46 depicts a flowchart for switching access by a service provider in an example of the invention. In this embodiment, the service provider selects a toll free premiere Internet service for a user using standard Internet access. The service provider pays for the changes in the toll free premium Internet service instead of the user—similar to toll free phone calls.

FIG. 46 begins in step 4600. In step 4602, the service application in the network device 562 waits for a request for a service. The user through the network device 512 transmits a request using a standard Internet access through the service network 530 for the service application in the network device 562 in step 4604. The service application in the network device 562 receives and processes the request. The service application then selects a toll free premiere quality of service (QOS) Internet access to the network device 512 in step 4606. The service application in the network device 562 generates and transmits an authentication instruction to the database system 522 in step 4608. The database system 522 receives and processes the authentication instruction. In step 4612, the database system 522 then generates and transmits a premium access instruction to establish the toll free premium Internet access between the access server 524 and the access server 554 via the service network 540.

Once the toll free premium Internet access is established, the user and service application exchange packets using the premium Internet access via the network device 512, the access server 524, the service network 540, the access server 554, and the network device 562 in step 4612. Once the service is completed, the database system 522 then generates and transmits a disconnect instruction to the access server 524, the service network 540, and the access server 554 to tear down the toll free premium Internet access in step 4614. The database system 522 then generates and transmits usage and performance statistics for the service application in the network device 562 in step 4616. Once the service is completed, the service application in the network device 562 returns the control to the user in step 4618 to select another service application in step 4602. In one embodiment, the database system 522 uses the cross connection system 583 to perform the operations disclosed in FIG. 46. In another embodiment, a third party such as the government performs the switching for surveillance or monitoring purposes.

Dynamic Switching Access

FIGS. 5 and 47 disclose one embodiment for dynamic switching access in an example of the invention. In a prior solution, a single access link from the user network to the access server is linked to a dedicated network in a single access session. However, users need to switch between different networks such as Internet and Intranets without tearing down the existing access session. No other systems allow the access server to be controlled by the data stream the access server is passing. Analog modems monitor the character stream for an escape sequence. The escape sequence notifies the modem that the data is for modem control and not for application data. However, no prior systems have implemented this functionality for a packet data network.

FIG. 47 depicts a flowchart for dynamic switching access in an example of the invention. In this embodiment, a user during an access session may switch networks without having the access session torn down. The access server 524 receives a control instruction from the user to switch from one dedicated service network 530 to another dedicated service network 540. FIG. 47 begins in step 4700. In step

42

4702, the access server 524 waits for a packet. In this embodiment, a request is in the format of a packet. The access server 524 then receives and processes the packet from the network device 512 in step 4704. The access server 524 checks if the packet is encoded for access control in step 4706. If the packet is not encoded for access control, the access server 524 processes the packet using standard access logic in step 4714 before returning to step 4702.

If the packet is encoded for access control, the access server 524 identifies the physical access path characteristics in step 4708. The access server 524 then checks if the access control is allowed in step 4710. If access control is not allowed, the access server 524 registers and logs an illegal access event in step 4716. The access server 524 then discards the packet in step 4718 before returning to step 4702.

If access control is allowed, the access server 524 generates and transmits an access control instruction to the database system 522 in step 4712. The database system 522 then receives and processes the access control instruction. The database system 522 identifies, authenticates, and authorizes the user and the requesting access server using the packet and path in step 4720. The database system 522 then retrieves the user access profile and the network device profile in step 4722. The database system 522 then checks if access control is allowed for the user and the network device based on the user access profile and the network device profile in step 4724. If access is not allowed, the database system 522 proceeds to step 4716.

If access is allowed, the database system 522 generates and transmits an access instruction to the access server 524 to update the access logic to switch to the service network 540 in step 4726. The database system 522 logs the access change in step 4727. In one embodiment, the database system 522 logs the access change in the audit system 572. The database system 522 also generates and transmits a reply with the new access status to the network device 512 in step 4728 before returning to step 4702. In one embodiment, the database system 522 uses the cross connect system 583 to perform the operations disclosed in FIG. 47.

Access Network Destination Control—FIGS. 5, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, and 59

Network Failover

Network devices become unavailable for a various reasons such as busy, failure, or overload. For network failover, when a destination network device is unavailable, network device requests are manually re-routed to a secondary network device. In order to avoid this manual rerouting, one prior art solution is the sharing of a network address between multiple processors in a box or multiple machines in a cluster. However, the requirement of destination and secondary devices residing in the same box or cluster makes this solution unacceptable for large systems.

Another prior art system is Network Address Translation, which translates a private network address to a public network address for a period of time. This system does not provide the capability of a network address to represent multiple failover destinations. Another solution is the Common Object Request Broker Architecture (CORBA), which provides the user a list of potential "logical" network devices to post the network device request. However, the user must implement CORBA interfaces and interactively select back up processes. The transaction context is lost when the primary fails. The user must re-establish the context. Unfortunately, none of these solutions provide an automatic translation from an unavailable destination network device to a secondary network device without operator

or user intervention, where the destination and secondary devices are not within the same box or cluster.

FIGS. 5, 48 and 49 disclose one embodiment for handling network address failover in an example of the invention. In a failover scenario, a destination network device fails or overloads, which makes the destination network device unavailable, and a secondary network device replicates the destination network device to give the user a fault tolerant appearance of the destination network device. Requests for the destination network device are translated to the secondary network device.

FIG. 48 depicts a flowchart for the access server 554 for network address failover in an example of the invention. FIG. 48 begins in step 4800. In step 4802, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 4804. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 4806, the access server 554 detects that the destination network device 562 fails and checks if the destination network address is enabled for failover in step 4806. If the destination network address is not enabled for failover, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 4812 before returning to step 4802.

If the destination network address is enabled for failover, the access server 554 checks if an address translation is active for the destination network device 562 in step 4808. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 4812 before returning to step 4802. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for failover in step 4810. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 4812 before returning to step 4802.

FIG. 49 depicts a flowchart for the database system 522 for network address failover in an example of the invention. FIG. 49 begins in step 4900. In step 4902, the database system 522 waits for a state event message or a timer event from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or a network device timer expires in step 4904. The database system 522 checks if the network device timer expired in step 4906.

If the network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout in step 4908. The database system 522 then retrieves and updates and the secondary network device's 564 profile to indicate that the destination network device 562 is unavailable in step 4910. In step 4912, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer has not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 4914. In step 4916, the database system 522 checks if the destination network device 562 is known. If the destination network device 562 is not known, the database system 522 generates

and transmits an unknown resource event before returning to step 4902. If the destination network device 562 is known, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics in step 4918. The database system 522 also resets the network device timer. The database system 522 then checks if the state event message is a failure or start event in step 4920.

If the state event message is not a failure or start event, the database system 522 checks if the load event threshold for the destination network device 562 is reached in step 4930. If the load event threshold is not reached, the database system 522 returns to step 4902. If the load event threshold is reached, the database system 522 proceeds to step 4926.

If the state event message is a failure or start event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 4922. In step 4924, the database system 522 checks if the state event message is a failure event. If the state event message is not a failure event, the database system 522 generates new active device profiles and then returns to step 4902. If the state event message is a failure event, the database system 522 proceeds to step 4926.

In step 4926, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address in step 4932 before returning to step 4902. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event in step 4928 before returning to step 4902. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 49.

Network Busy Forwarding

In a Public Switched Telephone Network (PSTN), a busy signal is sent to the caller when the called number destination is busy and thus unavailable. The caller can then select alternate actions such as delaying, retrying, or re-routing the request. Unfortunately, the PSTN handling of busy calls has not been applied to network devices in a data network. FIGS. 5, 50 and 51 disclose one embodiment for network busy forwarding in an example of the invention. When a destination network device 562 is busy, an access server 554 forwards the packets to a secondary network device 564. Requests for the destination network device are translated to the secondary network device.

FIG. 50 depicts a flowchart for the access server 554 for network busy forwarding in an example of the invention. FIG. 50 begins in step 5000. In step 5002, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5004. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5006, the access server 554 checks if the destination network device 562 is busy. If the destination network device 562 is not busy, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5012 before returning to step 5002.

If the destination network device 562 is busy, the access server 554 checks if an address translation is active for the destination network device 562 in step 5008. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step

5012 before returning to step 5002. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for failover in step 5010. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 5012 before returning to step 5002.

FIG. 51 depicts a flowchart for the database system 522 for network busy forwarding in an example of the invention. FIG. 51 begins in step 5100. In step 5102, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or the network device timer expired in step 5104. The database system 522 checks if a network device timer expired in step 5106.

If a network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout such as the network device being busy or unavailable in step 5108. The database system 522 then retrieves and updates and the secondary network device's 564 profile to indicate that the destination network device 562 is busy or unavailable in step 5110. The database system 522 also resets the network device timer. In step 5112, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer is not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 5114. In step 5116, the database system 522 checks if the destination network device 562 is known. If the destination network device 562 is not known, the database system 522 generates and transmits an unknown resource event before returning to step 5102. If the destination network device 562 is known, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics in step 5118. The database system 522 then checks if the state event message is a failure/busy or start/available event in step 5120.

If the state event message is not a failure/busy or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5130. If the load event threshold is not reached, the database system 522 returns to step 5102. If the load event threshold is reached, the database system 522 proceeds to step 5126.

If the state event message is a failure/busy or start/available event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 5122. In step 5124, the database system 522 checks if the state event message is a failure/busy event. If the state event message is not a failure/busy event, the database system 522 generates an address translation clear request in step 5132 before returning to step 5102. The address translation clear request adds a new active device profile or clears the existing translation. If the state event message is a failure/busy event, the database system 522 proceeds to step 5126.

In step 5126, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address if a busy

translation exists in the destination network device's 562 profile in step 5134 before returning to step 5102. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5128 before returning to step 5102. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 51.

Busy Flag

FIGS. 5 and 52 disclose one embodiment for a busy flag when the destination network device is busy in an example of the invention. When a destination network device 562 is busy and thus unavailable, an access server 554 replies to the user with a busy flag. FIG. 52 depicts a flowchart for the access server 554 for a busy flag when the destination network device is busy in an example of the invention. In one embodiment, the busy flag is a busy screen in HyperText Markup Language. FIG. 52 begins in step 5200. In step 5204, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5204. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5206, the access server 554 checks if the destination network device 562 is busy. If the destination network device 562 is not busy, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5210 before returning to step 5202. If the destination network device 562 is busy, the access server 554 replies to the user with a busy flag in step 5208 before returning to step 5202.

Timeout

FIGS. 5 and 53 disclose one embodiment for forwarding if the destination network device times out in an example of the invention. In this embodiment, a destination network device 562 receives a request packet from a user but fails to reply to the user within a pre-determined time. The failure to reply within a pre-determined time indicates the destination network device 562 is unavailable. The access server 554 then redirects the packet to a secondary network device 564 to handle forwarding when the destination network device fails to respond with the pre-determined time.

FIG. 53 depicts a flowchart for the access server 554 for forwarding if the destination network device times out in an example of the invention. FIG. 53 begins in step 5300. In step 5302, the access server 554 waits for the next packet or event. The access server 554 then checks whether a reply timeout event is received in step 5304. If a reply timeout event is not received, the access server 554 then receives and processes the packet in step 5306. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5308, the access server 554 checks if an address translation is available for the destination network device 562. If an address translation is not available for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5318 before returning to step 5302. If an address translation is available for the destination network device 562, the access server 554 sets a reply timer and caches the packet for the secondary network device 564. The access server 554 then transmits the packet with standard access logic to the destination network device 564 in step 5318 before returning to step 5302.

In step 5312, the access server 554 receives the reply timeout event and retrieves the cached request. The access server 554 then processes the reply timeout event and the cached request in step 5314. In step 5315, the access server

554 then replaces the packet's destination network address with the cached secondary network address. The access server 554 then transmits the packet to the secondary network device 564 with the standard access logic in step 5318 before returning to step 5302. In one embodiment, upon reply, the access server 554 deletes the cached packet copy.

Scheduled Alias Resolution

Network devices become unavailable for various reasons such as busy, failure, or overload. Also, service provider need to utilize network resources to improve efficiency and avoid network device latency. In voice telephony signaling networks, an SS7 SCP or an automatic call distributor (ACD) distributes calls to end user call centers. The SCP receives a 800 TCAP query and performs a database lookup to translate the 800 telephone number into a destination telephone number. The SCP may use the requester information and the time of day to perform the database lookup. Some problems with this system are SS7 signaling must be used and the system is limited by voice network constraints. Also, the phone numbers only allow numeric numbers for aliases. In data networks, prior art known as Object Request Brokers (ORB) in a Common Object Request Broker Architecture (CORBA) advertise process resources to a requester. The ORB also registers multiple instances of the same type of process resource.

FIGS. 5, 54, and 55 disclose one embodiment for scheduled alias resolution in an example of the invention. In this embodiment, an alias, domain name/local name, network address, or macro for a network resource is translated to another alias for another network resource based on a configurable schedule. Advantageously, load balancing of network devices is improved by scheduling different alias resolutions for different times/conditions. One problem with CORBA is the requester and resources must implement CORBA interfaces. Scheduled alias resolution is compatible with existing data network DNS implementations. Also, the load balancing, security, and failover may be based on the user access profile and the network device state.

FIG. 54 depicts a flowchart for the access server 554 for scheduled alias resolution in an example of the invention. FIG. 54 begins in step 5400. In step 5402, the access server 554 waits for the next request. The access server 554 then receives and processes a request from the network device 512 in step 5404. In this embodiment, the information including an alias is in the form of a request. In step 5406, the access server 554 checks if a current translation for the alias exists. If a current translation for the alias exists, the access server 554 translates the alias to the appropriate network address for a network device in user network 560 in step 5408. The access server 554 then replies to the network device 512 in step 5410 before returning to step 5402.

If a current translation for the alias does not exist, the access server 554 checks if a known access profile exists in the database system 522 exists for this user in step 5412. If a known access profile does not exist in the database system 522 for the user, the access server 554 generates and transmits an alias resolution failure message to the network device 512 in step 5414 before returning to step 5402. If a known access profile does exist in the database system 522 for the user, the access server 554 sets an external request timer in step 5416. The access server 554 then generates and transmits an alias translation request to the database system 522 in step 5418. The access server 554 then checks if a reply is received or the external request timer is expired in step 5420. If the reply is not received or the external request timer is not expired, the access server 554 returns to step

5420. If the reply is received or the external request timer is expired, the access server 554 checks if the reply was valid in step 5422. If the reply is invalid, the access server 554 returns to step 5414. If the reply is valid, the access server 554 then caches the alias translation for this user with time to live parameters in step 5424 before returning to step 5408.

FIG. 55 depicts a flow chart for the database system 522 for scheduled alias resolution in an example of the invention. FIG. 55 begins in step 5500. In step 5502, the database system 522 waits for a request. The database system 522 then receives and processes the request from the access server 554 in step 5504. In step 5506, the database system 522 then checks if the access server requester is known in step 5506. If the access server requester is not known, the database system 522 then registers an unknown requester event in step 5508. The database system 522 appends the reply to the packet and transmits the reply to the access server 554 in step 5510. The database system 522 replies with a decline message to the access server 554 in step 5512 before returning to step 5502.

If the access server requester is known, the database system 522 then checks whether the request is a translation appeal in step 5514. If the request is not a translation appeal, the database system 522 checks if the request is a security event in step 5516. If the request is a security event, the database system 522 increments a path/device security record in step 5522 before returning to step 5512. If the request is not a security event, the database system 522 registers an unknown event in step 5530 before returning to step 5502. If the request is a translation appeal, the database system 522 identifies and authenticates the user and the network device 512 in step 5518. If the access is not valid, the database system 522 returns to step 5522. If the access is valid, the database system 522 retrieves the user and network device's 512 profiles in step 5524. The database system 522 then generates the translation logic and calculates the translation time to live in step 5526. The database system 522 then appends the translation logic and the translation time to a reply to the access server 554. The database system 522 then replies with an approve message to the access server 554 in step 5528.

Forwarding

In a PSTN, a person may forward calls from their original phone number to another phone number where the person may be reached. Unfortunately, the PSTN handling of call forwarding has not been applied to network devices in a data network. FIGS. 5, 56 and 57 disclose one embodiment for destination controlled forwarding in an example of the invention. An access server 554 forwards the packets for a destination network device 562 that is unavailable to a secondary network device 564. Requests for the destination network device 562 are translated to the secondary network device 564.

FIG. 56 depicts a flowchart for the access server 554 for destination controlled forwarding in an example of the invention. FIG. 56 begins in step 5600. In step 5602, the access server 554 waits for the next packet from either the service network 530 or the service network 540. The access server 554 then receives and processes the packet in step 5604. In this embodiment, the access server 554 receives and processes information in the form of a packet. The access server 554 then checks if the packet is a reply in step 5606. If the packet is a reply, the access server 554 retrieves and deletes the cached destination network address in step 5608 before proceeding to step 5614.

If the packet is not a reply, the access server 554 checks if an address translation is active for the destination network

device 562 in step 5610. If an address translation is not active for the destination network device 562, the access server 554 transmits the packet with standard access logic to the destination network device 562 in step 5614 before returning to step 5602. If an address translation is active for the destination network device 562, the access server 554 processes the packet to replace the destination network address with the secondary network address that is cached for the destination network address for destination controlled forwarding in step 5612. The access server 554 then transmits the packet with standard access logic to the secondary network device 564 in step 5614 before returning to step 5602.

FIG. 57 depicts a flowchart for the database system 522 for destination controlled forwarding in an example of the invention. FIG. 57 begins in step 5700. In step 5702, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives a state event message from the destination network device 562 or a network device timer expires in step 5704.

The database system 522 processes the state event message and authenticates the destination network device 562 in step 5706. In step 5708, the database system 522 checks if the access server requester is known. If the access server requester is not known, the database system 522 generates and transmits an unknown requester event before returning to step 5702. If the access server requester is known, the database system 522 retrieves and updates the access server's 554 profile with availability and performance statistics in step 5710. The database system 522 then checks if the state event message is a failure/forward or start/available event in step 5712.

If the state event message is not a failure/forward or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5716. If the load event threshold is not reached, the database system 522 returns to step 5702. If the load event threshold is reached, the database system 522 proceeds to step 5722.

If the state event message is a failure/forward or start/available event, the database system 522 retrieves and updates the secondary network device's 564 profile in step 5718. In step 5720, the database system 522 checks if the state event message is a failure/forward event. If the state event message is not a failure/forward event, the database system 522 generates and transmits an address translation clear request message to the access server 554 in step 5726 before returning to step 5702. If the state event is a start event, then a new availability profile is generated. If the state event message is a failure/busy event, the database system 522 proceeds to step 5722.

In step 5722, the database system 522 checks if the secondary network device 564 is available. If the secondary network device 564 is available, the database system 522 generates and transmits an address translation request message to the access server 554 to change packets' destination network address to the secondary network address if a busy translation exists in the destination network device's 562 profile in step 5728 before returning to step 5702. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5724 before returning to step 5702. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 57.

Network Endpoint Availability Management

FIGS. 5, 58, and 59 disclose one embodiment for network endpoint availability management in an example of the invention. The features of failover, busy, busy flag, timeout, and forwarding are combined into this embodiment. FIG. 58 depicts a flowchart for the access server 554 network endpoint availability management in an example of the invention. FIG. 58 begins in step 5800. In step 5802, the access server 554 waits for the next packet or event. The access server 554 then checks whether a reply timeout event is received in step 5804.

If a reply timeout event is not received, the access server 554 then receives a data packet and processes the packet in step 5806. In this embodiment, the access server 554 receives and processes information in the form of a packet. In step 5808, the access server 554 checks if the packet is a reply. If the packet is a reply, the access server 554 clears the cached request and clears a reply timer in step 5810 before returning to step 5804. If the packet is not a reply, the access server 554 checks if an address translation is active for the destination network device 562 in step 5812. If an address translation is active for the destination network device 562, the access server 554 proceeds to step 5824. If an address translation is not active for the destination network device 562, the access server 554 checks if the destination address is busy. If the destination address is not busy, the access server 554 proceeds to step 5830. If the destination address is not busy, the access server 554 checks if a forward address translation is available in step 5814. If the forward address translation is not available, replies to the user with a busy flag in step 5818 before returning to step 5802. If the forward address translation is available, the access server 554 proceeds to step 5824.

If a reply timeout event is received, the access server 554 receives the reply timeout event and the cached request in step 5820. The access server 554 then processes the reply timeout event and the cached request in step 5822. In step 5824, the access server 554 then replaces the packet's destination network address with the cached secondary network address. The access server 554 then transmits the packet to the secondary network device 564 with the standard access logic in step 5824.

In step 5826, the access server 554 checks if the forward address is busy. If the forward address is busy, the access server 554 returns to step 5818. If the forward address is not busy, the access server 554 sets a reply timer and caches the request in step 5828. In step 5830, the access server 554 then transmits the packet with standard access logic before returning to step 5802.

FIG. 59 depicts a flowchart for the database system 522 for network endpoint availability management in an example of the invention. FIG. 59 begins in step 5900. In step 5902, the database system 522 waits for a state event message from a network device in the user network 560. The database system 522 receives the state event message from the destination network device 562 or the network device timer expired in step 5904. The database system 522 checks if the network device timer expired in step 5906.

If the network device timer expired, the database system 522 retrieves and updates the destination network device's 562 profile with availability and performance statistics to reflect a network device timeout such as the network device being busy or unavailable in step 5908. The database system 522 then retrieves and updates the secondary network device's 564 profile to indicate that the destination network device 562 is busy or unavailable in step 5910. The database system 522 also resets the network device timer. In step 5912, the database system 522 generates and transmits an

51

address translation request message to the access server 554 to change packets' destination network address to the secondary network address.

If the network device timer has not expired, the database system 522 processes the state event message and authenticates the destination network device 562 in step 5914. In step 5916, the database system 522 checks if the access server requester is known. If the access server requester is not known, the database system 522 generates and transmits an unknown resource event before returning to step 5902. If the access server requester is known, the database system 522 retrieves and updates the access server's 554 profile with availability, activity logs, and performance statistics in step 5918. The database system 522 then checks if the state event message is a failure, busy, forward or start/available event in step 5920.

If the state event message is not a failure, busy, forward or start/available event, the database system 522 checks if the busy load event threshold for the destination network device 562 is reached in step 5930. If the load event threshold is not reached, the database system 522 returns to step 5102. If the load event threshold is reached, the database system 522 proceeds to step 5926.

If the state event message is a failure, busy, forward, or start/available event, the database system 522 retrieves and updates the destination network device 562 and secondary network device's 564 profile in step 5922. In step 5124, the database system 522 checks if the state event message is a failure, busy, or forward event. If the state event message is not a failure/busy event, the database system 522 generates and transmits an address translation clear request to the access server 554 if a busy translation exists in step 5934 before returning to step 5902. If the state event message is a failure, busy, or forward event, the database system 522 proceeds to step 5926.

In step 5926, the database system 522 checks if a secondary network device 564 is available for forwarding. If a secondary network device 564 is available, the database system 522 checks if the state event message is a failure or busy event in step 5936. If the state event message is a failure or busy event, the database system 522 generates and transmits a busy address translation request message to the access server 554 in step 5932 before returning to step 5902. If the state event message is not a failure or busy event, the database system 522 generates and transmits a forward address translation request message to the access server 554 to change packets' destination network address to the secondary network address in step 5938 before returning to step 5902. If the secondary network device 564 is not available, the database system 522 registers a no secondary network device failure event and transmit a busy message to the user in step 5928 before returning to step 5902. In one embodiment, the database system 522 uses the availability database system 594 to perform the operations disclosed in FIG. 59.

Access Network Audit Server—

FIGS. 5, 60, 61, 62, and 63

Service Capability Monitor

FIGS. 5, 60, 61, 62, and 63 disclose one embodiment for service capability monitor in an example of the invention. In a prior system, Java applet technology and Sun's JINI project delivers communication and state mechanisms from the function provider to the function requesters. Java applet technology allows a mechanism for device control without any coordinating control agent. JINI provides a registry which maintains capabilities like a CORBA ORB as a means for primitive control. One problem is that JINI focuses on

52

the distribution of mechanisms instead of the interworking of mechanisms in a standardized way. In this embodiment of service capability control, a network access provider monitors the validity and state of the application level request to a network service provider. The network access provider monitors performance of their network and the service provider's network service requiring an understanding of the other provider's service requested. Access providers such as Internet service providers could mitigate their liability and increase the service's performance by extending their service view to the user base. Specific service environment access points may be monitored and service access rights and restrictions may be enforced. This embodiment provides an interworking mechanism that provides a dynamic interface to a service based network. The monitoring interface must be dynamic because different service have different potential states, different legal and illegal state transitions, and different communication mechanisms.

FIG. 60 depicts a flowchart for a firewall 526 for service capability monitor in an example of the invention. FIG. 60 begins in step 6000. In step 6002, the firewall 526 waits for a packet. The firewall 526 then receives and processes a packet from the network device 512 in step 6004. In step 6006, the firewall 526 performs standard firewall functions. The firewall 526 then copies the packets and transmits the packet to the destination and the database system 522 in step 6008 before returning step 6002.

FIG. 61 depicts a service capability monitor software architecture for a service capability monitor in an example of the invention. A service capability monitor software architecture 6100 comprises a service broker tokenizer process B 6112, a service broker state map process C 6114, a network A controller process A 6116, a service A signal tokenizer process E 6122, a service A state map process F 6124, a service A controller process D 6126, a service B signal tokenizer process H 6132, a service B state map process F 6134, and a service B controller process D 6136. The network A controller process A 6116 includes a service broker session state objects 6118. The service A controller process D 6126 includes a service A session state objects 6128. The service B controller process D 6136 includes a service B session state object 6138.

The network A controller process A 6116 is connected to the service broker tokenizer process B 6112, the service broker state map process C 6114, the service A controller process D 6126, and the service B controller process D 6136. The service A controller process D 6126 is connected to the service A signal tokenizer process E 6122 and the service A state map process F 6124. The service B controller process D 6136 is connected to the service B signal tokenizer process H 6132 and the service B state map process F 6134.

FIG. 62 depicts a flowchart for the network logic for service capability monitor in an example of the invention. FIG. 62 begins in step 6200. In step 6202, the database system 522 waits for the packet. The database system 522 then receives and processes the packet from the firewall 526 in step 6204. The database system 522 then identifies and authorizes the requester in step 6206. The database system 522 checks if the requester is known in step 6208.

If the requester is known, the network A controller process 6116 writes the packet to process B—service broker signal tokenizer process 6112 in step 6210. The process B 6112 then replies with the function token and parameters to the process A 6116 in step 6212. The database system 522 then checks if a service session state object (SSO) 6118 exists for this session in step 6214. If the SSO 6118 does not exist, the database system 522 creates a SSO 6118 for this session and

53

initializes the current state in step 6216. If the SSO 6118 exists, the database system 522 then proceeds to step 6218. In step 6218, process A 6116 writes the state token and current state to process C—service broker state map process 6114 in step 6218. The process C 6114 then replies with a transition token and parameters to the process A 6116 in step 6220. The process A 6116 applies the transition token and the parameters to the SSO 6118 to update the current state and setting actions in step 6222.

In step 6224, the database system 522 checks if there is an action indicated by the SSO 6118. If there is not an action indicated by the SSO 6118, the database system 522 proceeds to step 6228. If there is an action indicated by the SSO 6118, the database system 522 performs the transition action specified by the SSO 6118 in step 6226. In step 6228, the database system 522 updates the session sliding usage accumulators. The database system 522 then checks if there is an action in the SSO 6118 for this usage in step 6230. If there is no action, the database system 522 returns to step 6202. If there is an action in the SSO 6118, the database system 522 performs the usage action specified in the SSO 6118 before returning to step 6202.

If the requester is not known, the database system 522 checks if there is a network A state object in step 6234. If there is a network state object, the database system 522 proceeds to step 6238. If there is no network state object, the database system 522 creates a network A state object and initializes the current state in step 6236. In step 6238, the database system 522 increments network A state object usage for this event. The database system 522 then checks if there is an action in the network A state object for this usage in step 6240. If there is no action in the network A state object for this usage, the database system 522 returns to step 6202. If there is an action in the network A state object for this usage, the database system 522 performs the usage action specified in the network A state object in step 6242 before returning to step 6242.

FIG. 63 depicts a flowchart for the service logic for service capability monitor in an example of the invention. FIG. 63 begins in step 6300. In step 6302, process D service A controller process 6126 waits for the packet. The process D 6126 then receives and processes the packet from the process A network A controller 6116 in step 6304. The process D 6126 then identifies and authorizes the requester in step 6306. The process D 6126 checks if the requester is known in step 6308.

If the requester is known, the process D 6126 writes the request to process E service A signal tokenizer process 6122 in step 6310. The process E 6122 then replies with the state token and parameters to the process D 6126 in step 6312. The database system 522 then checks if a service session state object (SSO) 6128 exists for this session in step 6314. If the SSO 6128 does not exist, the database system 522 creates a SSO 6128 for this session and initializes the current state in step 6316. If the SSO 6118 exists, the database system 522 then proceeds to step 6318. In step 6218, the process D 6126 writes the state token and current state to process F—service state map process 6124 in step 6318. The process F 6124 then replies with a transition token and parameters to the process D 6126 in step 6320. The process D 6126 applies the transition token and the parameters to the SSO 6128 to update the current state and setting actions in step 6322.

In step 6324, the database system 522 checks if there is an action indicated by the SSO 6128. If there is not an action indicated by the SSO 6128, the database system 522 proceeds to step 6328. If there is an action indicated by the SSO

54

6128, the database system 522 performs the transition action specified by the SSO 6128 in step 6326. In step 6328, the database system 522 updates the session sliding usage accumulators. The database system 522 then checks if there is an action in the SSO 6128 for this usage in step 6330. If there is no action, the database system 522 returns to step 6302. If there is an action in the SSO 6128, the database system 522 performs the usage action specified in the SSO 6128 before returning to step 6302.

If the requester is not known, the database system 522 checks if there is a service A state object in step 6334. If there is a service state object, the database system 522 proceeds to step 6338. If there is no service state object, the database system 522 creates a service A state object and initializes the current state in step 6336. In step 6338, the database system 522 increments service A state object usage for this event. The database system 522 then checks if there is an action in the service A state object for this usage in step 6340. If there is no action in the service A state object for this usage, the database system 522 returns to step 6302. If there is an action in the service A state object for this usage, the database system 522 performs the usage action specified in the service A state object in step 6342 before returning to step 6342.

EXAMPLE

The following is one example of the access communication system that integrates many of the features described in the various embodiments of the inventions above. In the example below, a subscriber called Bank has many employees. The Bank has access to the network architecture 500 through an access provider called AccProv1. In this example, a bank employee is retrieving a stock quote from the yahoo.com website and then switches to request a streaming video from the yahoo.com website. Yahoo's web-server is running in the user network 562. Yahoo has access to the network architecture 500 through an access provider called AccProv2.

A system administrator for AccProv1 configures and optionally starts an access execution environment for the subscriber Bank in the local database system 570. The system administrator also starts a subscriber proxy for the Bank that runs in the access execution environment for the Bank. The subscriber proxy includes transport handlers, configuration handlers, and security handlers. The AccProv1 system administrator starts access execution environments for the maximum number of concurrent users in the local database system 570. The system administrator also starts generic proxies for users in the local database system 570. These generic proxies become user proxies after the user access profile is retrieved for the user.

The system administrator for AccProv2 starts an access execution environment for a service for Yahoo in the local database system 590. The system administrator also starts a service proxy for Yahoo's service that runs in the access execution environment for Yahoo. A system administrator for the Bank then sets up the user access profiles for each employee by inheriting attributes for the user access profile from capability classes or groups that the user belongs to. The user access profile includes the network shell for the user.

The Bank employee then logs on to the access server 524 using a user ID. The Bank system administrator set up. Besides user ID, a user may login to the access server 524 using a prepaid account code, a bank card number, an access card account code, or a user ID including a delimiter for user

access mobility. Upon connection, the access server 524 creates a device session in an access execution environment for the user. Then in response to the user login, the database system 522 retrieves the user access profile for the Bank employee. For a prepaid account code, a bank card number, an access card account code, and a user ID with a delimiter, the database system 522 may retrieve the user access profile from a database system external to the local database system 570. Also, the database system 522 may retrieve the user access profile from a database system external to the local database system 570 for global authentication.

Once the user access profile is retrieved, the access server 524 creates a user session in the access execution environment for the user and binds the device session with the user session. The configuration handler of the user proxy includes the network shell retrieved with the user access profile. The access server 524 then generates and transmits a list of available services in a service based directory to the Bank employee based on the user access profile.

The Bank employee then selects a service to check a stock quote for IBM by transmitting a request for service to the access server 524. The request for service may be in the form of a selection from the service based directory or an alias for alias translation for the network shell using domain name scoping or inbound DNS lookup. The access server 524 processes and transfers the request for service to the user proxy for the Bank employee. The user proxy transfers the request to the service proxy for Yahoo. The service proxy processes and transfers the request for service to Yahoo's web server. The database system 522 translates the service request for Yahoo to a private fulfillment address for one of Yahoo's web servers. The access server 554 then determines if a private destination address is available. If the private destination address is unavailable, the access server 554 could perform many actions including forwarding the request to another Yahoo web server or replying with a busy flag. The access server 524 also binds the user session and the device session with the selected service session by exchanging reference ID's.

The Bank employee then requests another stock quote for Oracle. The access server 524 then determines this stock quote was already cached based on the user's predictable pattern of requesting quotes for this company or a group of software companies. These prior requests for stock quotes of software companies by the Bank employee were audited to derive the Bank employee's predictable pattern. Alternatively, the request for the stock quote for Oracle could have been setup by the user in a script of commands to cache the quote in advance.

The Bank employee then requests a streaming video from Yahoo. The access server 524 switches the access for a premiere access based on the request for streaming video from the Bank employee. Alternatively, Yahoo initiates a switch of access for toll-free Internet service to the Bank employee who is a preferential customer. In order to be able to run the streaming video, the database system 522 also inherits user profile information from a class for streaming video user and updates the user access profile with the user profile information. The user proxy transfers the request for streaming video to the subscriber proxy. The subscriber proxy transfers the request to the service proxy. The service proxy then transfers the request to the Yahoo for one of their streaming video servers. Yahoo's streaming video server then provides the streaming video to the Bank employee.

Conclusion

The access network operates as described in response to instructions that are stored on storage media. The instruc-

tions are retrieved and executed by a processor in the access network. Typically, the processor resides in a server or database. Some examples of instructions are software, program code, and firmware. Some examples of storage media are memory devices, tape, disks, integrated circuits, and servers. The instructions are operational when executed by the processor to direct the network to operate in accord with the invention. The term "processor" refers to a single processing device or a group of inter-operational processing devices. Some examples of processors are computers, integrated circuits, and logic circuitry. Those skilled in the art are familiar with instructions, processors, and storage media.

Those skilled in the art will appreciate variations of the above-described embodiments that fall within the scope of the invention. As a result, the invention is not limited to the specific examples and illustrations discussed above, but only by the following claims and their equivalents.

1 claim:

1. A method of operating an access system including an access server to provide access between a user system and a plurality of communication networks that provide services to a user, the method comprising:

receiving a user login into the access server;

processing the user login to determine if the user is allowed access to the access system based on a local database system;

providing access to the access system to the user in response to the determination that the user is allowed access based on the local database system;

generating an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system;

in the second database system, receiving and processing the authorization query to determine whether the user is allowed access;

in the second database system, generating and transmitting an authorization response to the local database system;

receiving and processing the authorization response indicating whether the user is allowed to use the access system from the second database system; and

providing access to the access system to the user in response to the authorization response that allows the user to use the access system.

2. The method of claim 1 further comprising generating and transmitting a login query.

3. The method of claim 1 wherein processing the user login to determine if the user is allowed access to the access system based on a local database system is based on a user access profile.

4. The method of claim 1 further comprising determining if the second database system exists.

5. The method of claim 1 further comprising disconnecting a user's network device from the access server in response to the determination that the user is not allowed to use the access system.

6. The method of claim 1 wherein the login reply includes an access card account code.

7. The method of claim 1 wherein the login reply includes foreign network account information and further comprising identifying the second external database based on the foreign network account information.

8. The method of claim 1 further comprising:

receiving a request for access into a service control point;

processing the request for access to determine the destination for the request;

generating and transmitting a reply to route the request to the access server;

9. The method of claim 1 further comprising logging contract and settlement information.

10. An access system for providing access between a user system and a plurality of communication networks that provide services to a user, the access system comprising:

an access server connected to the user system and the plurality of communication networks and configured to receive and transmit a user logon from the user system to a local database system;

the local database system connected to the access server and configured to receive the user logon, process the user logon to determine if the user is allowed access to the access system based on the local database system, provide access to the access system to the user in response to the determination that the user is allowed access based on the local database system, generate an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system, receive and process an authorization response indicating whether the user is allowed to use the access system from the second database system, and provide access to the access system to the user in response to the authorization response that allows the user to use the access system; and

the second database system connected to the local database system and configured to receive and process the authorization query to determine whether the user is allowed access and generate and transmit the authorization response for the local database system.

11. The access system of claim 10 wherein the access server is configured to generate and transmit a logon query.

12. The access system of claim 10 wherein the local database system is configured to process the user logon to determine if the user is allowed access to the access system based on a user access profile.

13. The access system of claim 10 wherein the local database system is configured to determine if the second database system exists.

14. The access system of claim 10 wherein the access server is configured to disconnect a user's network device in response to the determination that the user is not allowed to use the access system.

15. The access system of claim 10 wherein the logon reply includes an access card account code.

16. The access system of claim 10 wherein the logon reply includes foreign network account information and the local database system is configured to identify the second external database based on the foreign network account information.

17. The access system of claim 10 further comprising:

a service control point connected to the user system configured to receive a request for access, process the request for access to determine the destination for the request, and generate and transmit a reply to route the request to the access server.

18. The access system of claim 10 wherein the local database system is configured to log contract and settlement information.

19. A software product for providing access between a user system and a plurality of communication networks that provide services to a user, the software product comprising:

database software operational when executed by a processor to direct the processor to receive the user logon, process the user logon to determine if the user is allowed access to an access system based on a local database system, provide access to the access system to the user in response to the determination that the user is allowed access based on the local database system, generate an authorization query for a second database system external to the local database system in response to the determination that the user is not allowed access based on the local database system, receive and process an authorization response indicating whether the user is allowed to use the access system from the second database system, and provide access to the access system to the user in response to the authorization response that allows the user to use the access system; and

a software storage medium operational to store the database software.

20. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to process the user logon to determine if the user is allowed access to the access system based on a user access profile.

21. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to determine if the second database system exists.

22. The software product of claim 19 wherein the logon reply includes an access card account code.

23. The software product of claim 19 wherein the logon reply includes foreign network account information and the database software is operational when executed by the processor to direct the processor to identify the second external database based on the foreign network account information.

24. The software product of claim 19 wherein the database software is operational when executed by the processor to direct the processor to log contract and settlement information.

* * * * *

U.S. PATENT DOCUMENTS

6,195,697 B1	2/2001	Bowman-Amuah	6,272,152 B1	8/2001	Levin et al.
6,208,720 B1	3/2001	Curtis et al.	6,282,276 B1	8/2001	Felger
6,212,262 B1	4/2001	Kamel	6,289,010 B1	9/2001	Voit et al.
6,233,313 B1	5/2001	Farris et al.	6,295,292 B1	9/2001	Voit et al.
6,247,047 B1	6/2001	Wolff	6,330,543 B1	12/2001	Kepecs
6,260,024 B1	7/2001	Shkedy	2002/0147658 A1	10/2002	Kwan

* cited by examiner

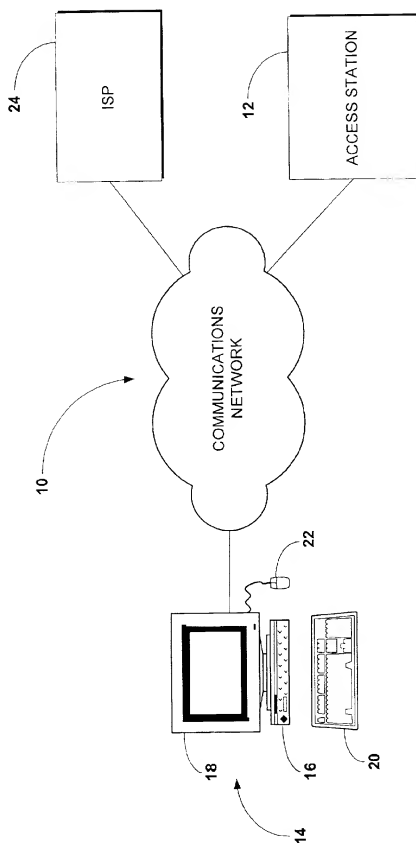


Figure 1.

Figure 2a.

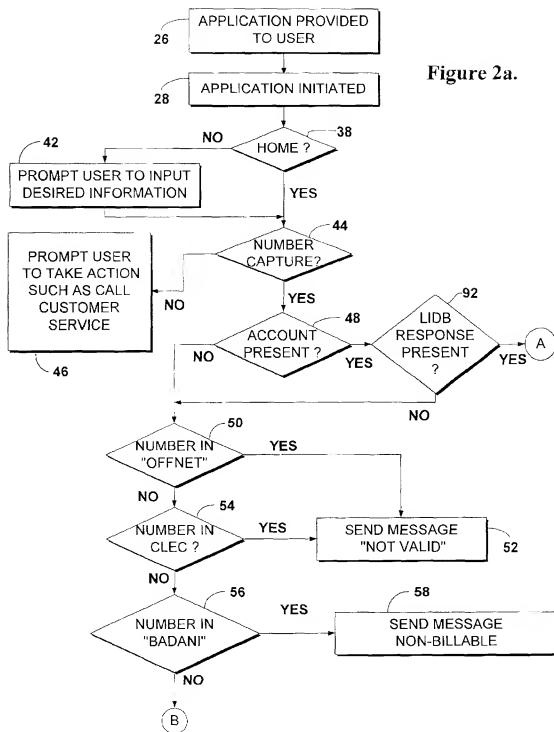
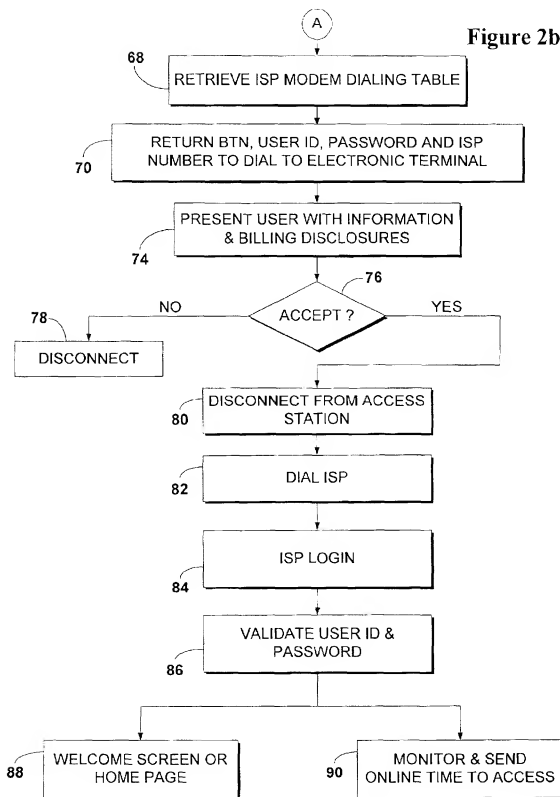
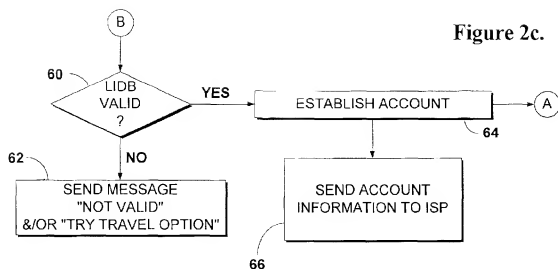


Figure 2b.





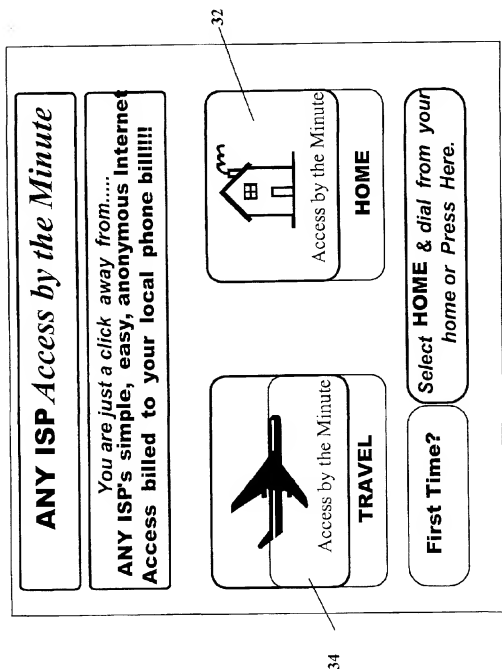


Figure 3a.

ANY ISP *Access by the Minute*

First Time??

Please ensure you:

- are dialing from your home phone
- are authorized to charge to this phone number
- understand this call will be charged per minute

CONTINUE

BACK

Figure 3b.

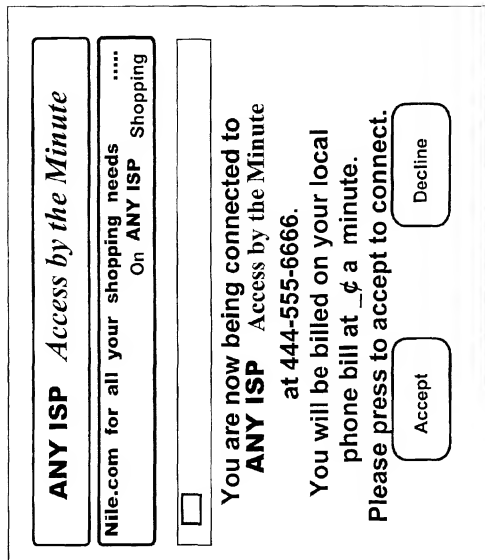


Figure 3c.

PACIFIC BELL

Account Number 408 229 765 N 6159 Statement Date Dec 3, 1998 Page 11
 Questions about your bill? 1-800-736-7900



Total Current Charges (See detail below)

Monthly Charges
 Description: 1.00 Local: 1.00 Amount: 1.60
 Total Monthly Charges: 1.60

Calls from 408 XXX-XXXX Billed on Behalf of AOL Access By The Minute Direct Dialed ISP Access Calls:

Date	Time	Place and Number Called	Type	Day	Minutes	Amount
6	Feb 10	2:17pm Billing: MT	Direct	Day	6	.42
7	Feb 11	12:30pm Billing: MT	Direct	Day	10	.70
8	Feb 11	12:30pm Billing: MT	Direct	Night	7	.48
9	Feb 14	12:30pm Billing: MT	Direct	Day	8	.56
10	Feb 16	2:56pm Billing: MT	Direct	Day	8	.56
11	Feb 17	3:46pm Billing: MT	Direct	Day	13	.84
12	Feb 22	3:46pm Billing: MT	Direct	Day	5	.36
13	Feb 23	12:30pm Billing: MT	Direct	Day	5	.36
14	Feb 23	8:34pm Billing: MT	Direct	Day	5	.36
15	Feb 23	8:34pm Billing: MT	Direct	Day	5	.36
Total ISP Access Calls					66	\$16.90

Taxes & Surcharges

Description	Amount
16 Charges for Network Access for Interstate Calling	3.00
17 CA High Cost Fund Surcharge A	.07
18 CA High Cost Fund Surcharge B	.07
19 Rate Surcharge	1.34
20 State Regulatory Fee	.03
21 State Regulatory Fee	.03
22 Tax, Fed, State, 99 011 Local	1.50
Total Taxes & Surcharges	\$4.31

13 6291 B251 1A 408 2244667 765 9613818E7 R001 RTEN 9273

Figure 4.

1

SYSTEM AND METHOD FOR ACCESSING THE INTERNET ON A PER-TIME-UNIT BASIS

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention is generally directed to a system and method for accessing a global electronic communications network. More particularly, the present invention is directed to a system and method for accessing the Internet from an electronic terminal, wherein an account associated with the electronic terminal is billed in monetary units corresponding to the length of time the electronic terminal is connected to the Internet.

2. Description of the Related Art

Internet access has conventionally required users to obtain a subscription from an Internet service provider. The form of such subscriptions vary, depending upon the provider and the quantity of internet access desired. For example, one conventional approach is to charge a subscriber a fixed monetary amount for unlimited use of the Internet, via the provider, during a predetermined period, such as a month. In variations of this approach, a user may be charged a first fixed amount for a first amount of access time during a given period, and then additional amounts for access over the initially allotted time. Charges for Internet access are typically billed to a financial account, such as a credit card of the subscriber.

One drawback to conventional approaches for providing an Internet access is the requirement is that the user establish a subscription and, particularly, the requirement that a user enter financial account information in order to establish the subscription. While electronic transactions over the Internet are becoming increasingly common place, a significant number of individuals, and especially those that may remain skeptical of security issues or, alternatively, those that simply have had little or no exposure to the Internet, may remain unwilling to submit financial information to an Internet service provider in order to establish the subscription. Additionally, those users that access the Internet infrequently or sporadically may be unwilling to incur periodic subscription charges.

Accordingly, the need exists for a system and method for accessing the Internet that provides an alternative to existing Internet access arrangements. In particular, the need exists for a system and method for accessing the Internet in a manner which does not require a conventional subscription. The present invention fulfills these and other needs.

SUMMARY OF THE INVENTION

The present invention is a system and method for accessing global electronic communications network and, particularly, for accessing the Internet. The system of the present invention has an Internet access station, and a site associated with an Internet service provider. Using an electronic terminal, such as a personal computer, or a fixed, portable, or mobile computing station, a user may access the Internet via the Internet service provider, albeit in accordance with the unique aspects of the present invention.

In accordance with one aspect of the present invention, the access station is a site at which various processes are performed as a precondition to permitting the electronic terminal to establish a communications link with the Internet service provider. Additionally, the access station also pref-

2

erably serves as a billing station, such that charges associated with Internet access with the electronic terminal, through the Internet service provider, are billed to the user of the electronic terminal via the access station.

In particular, in accordance with the principles of the present invention, application software for permitting the user of the electronic terminal to access the Internet are provided to the electronic terminal. The application software permits a user of the electronic terminal to establish a communications link with the access station. Once a communications link with the access station is established, the access station, utilizing a processor, processes a series of steps in order to automatically establish an account associated with the electronic terminal. Once the account is established, information indicative of the account is transmitted via the communications link back to the electronic terminal along with connection information, such as a telephone number, for enabling the electronic terminal to initiate a second communications link with the Internet service provider. Additionally, information indicative of the established account is transmitted via a third communications link from the access station directly or at scheduled intervals to the Internet service provider, where it is stored as a profile.

Upon receipt by the electronic terminal of the account information and connection information, the first communications link between the electronic terminal and the access station is terminated, and the electronic terminal (using the connection information received from the access station) initiates a second communications link with the Internet service provider. Once the second communications link between the electronic terminal and Internet service provider is established, the electronic terminal transmits the account information received from the access station to the Internet service provider. The Internet service provider processes that information by comparing it with the short profile information received from the access station, in order to confirm that the electronic terminal seeking access is a valid terminal for which access should be permitted. Upon completion of that validation process, the electronic terminal is logged on to the Internet via the Internet service provider.

In accordance with an aspect of the present invention, upon initiation of the second communications link between the electronic terminal and the Internet service provider or, at some other convenient time, such as upon a log on time to the Internet, the Internet service provider begins monitoring the time of the second communications link (or log on time). That time duration is monitored in time units, such as minutes, and is stored in an electronic memory of the Internet service provider for periodic transmission directly to the access station. It will be understood and appreciated that while the time of the communications link or Internet access time is preferably transmitted from the Internet service provider to the access station periodically, it may be continuously transmitted via a direct communications link between the Internet service provider and access station. At the access station, the time units associated with the second communications link, or log in time, is associated with the account established for the electronic terminal, and a monetary rate is multiplied by the time units for billing purposes. In particular, although the monetary charges may be calculated at the Internet service provider, these calculations may be made at the access station by multiplying the number of monetary units associated with a particular log in session by a published rate, and associating the resulting charges with the established account.

In accordance with a particular aspect of the present invention, the access station automatically determines the

3

telephone number from which the electronic terminal is accessing the access station. In a preferred embodiment, the determined telephone number is utilized as a billing number in the process of establishing the account. Thus, in accordance with the invention, any charges associated with Internet access per time unit are billed to an account or subscription associated with the telephone number from which the electronic terminal accesses the access station. Particularly, the charges are consolidated on a telephone bill which includes other charges corresponding to the telephone number, such as local and long distance telephone calls.

Additionally, before permitting the electronic terminal to link with the Internet service provider, the access station preferably performs various validation processes for determining whether the telephone number associated with the electronic terminal is acceptable for billing purposes. For example, the access station preferably determines whether the telephone number is within one or more selected networks, and determines whether there are other acceptable credit risks associated with the telephone number. Additionally, the access station preferably retrieves information from a line information data base (LIDB) to ascertain whether the telephone number is associated with a valid, existing telephone line subscription. These validation features are provided upon initial access by the electronic terminal to the access station, but are preferably only provided on subsequent Internet access attempts when the time lapse between access attempts is greater than a predetermined period, such as for example, 30 days.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects and features of the invention noted above are explained in more detail with reference to the drawings, in which like reference numerals denote like elements, and in which:

FIG. 1 is a block diagram of the Internet access system of the present invention;

FIGS. 2a-2c illustrate a flow chart illustrating the method of the present invention and the software processes performed by the present invention;

FIGS. 3a-3c are examples of screen displays provided on the display screen of the electronic terminal utilized for accessing the Internet in accordance with the principles of the present invention; and

FIG. 4 is an example of monetary charges incurred for accessing the Internet in accordance with the principles of the present invention, wherein those charges are consolidated on a standard telephone bill.

DETAILED DESCRIPTION OF THE INVENTION

With reference initially to FIG. 1, an Internet access system of the present invention is denoted generally by reference numeral 10. The Internet access system 10 of the present invention has an Internet access station, denoted by reference numeral 12, which serves a variety of functions, as described in detail below. In particular, Internet access station 12 is preferably a site on a global communications network at which an Internet access account is established and maintained, at which Internet access validation procedures are conducted and at which various billing activities are processed. The Internet access system 10 further includes an electronic terminal 14. Electronic terminal 14 is illustrated in FIG. 1 as a personal computer having a processor 16, a display 18, a keypad input 20, and a mouse 22. It will

4

be understood and appreciated that the electronic terminal 14 may include other desired components, such as microphone input, speakers, a scanner, a web camera, etc. Additionally, it should be understood and appreciated that the electronic terminal 14, rather than being a conventional personal computer, may be any type of electronic terminal for accessing the Internet. For example, the electronic terminal 14 may be a portable or laptop computer, or a mobile or wireless Internet equipped communications device, such as a cellular telephone.

Internet access system 10 further includes an Internet service provider, denoted generally by reference numeral 24. Internet service providers are well known. In particular, Internet service provider 24 is a site, at an address, on a global communications network. In particular, Internet service provider 24 is a site which serves as a portal through which users of electronic terminals may access the Internet. As illustrated, a communications network is provided for establishing communications links between electronic terminal 14, access station 12, and Internet service provider 24. The communications network may be any conventional type of network, such as a telephony network, or a data network, may be a broadband (or not) network, etc. or any combination thereof. Access to the network, through the various components of the present invention, may be made in a conventional manner through interface components, such as a modem or transceiver.

With additional reference to FIGS. 2a-2c, the system 10 and its method of operation will be described in detail.

In accordance with an aspect of the present invention, and as indicated at step 26 in FIG. 2a, an Internet access software application is provided to electronic terminal 14 in any conventional manner. For example, a user of electronic terminal 14 may download the necessary application software from access station 12 or Internet service provider 24. Alternatively, the application software may be provided on a transferable medium, such as a CD, with which the electronic terminal 14 communicates. Thus, as will be understood and appreciated, the application software may be stored on either a portable, transferable medium, or may be stored on a hard drive memory within the electronic terminal 14. Alternatively, the Internet access application software of the present invention may be provided in an open system arrangement, in which the Internet access software of the invention is provided at the access station 12, such that electronic terminal 14 communicates with the application software via the communications network, although the application software is not physically stored at the electronic terminal 14.

As indicated at reference numeral 28, a user of electronic terminal 14 initiates the Internet application software of the present invention in any conventional manner, such as by utilizing computer mouse 22 to click on an icon associated with the software displayed on display screen 18. Upon initiation of the application software at step 28, electronic terminal 14 displays at display 18 a display screen, such as that illustrated in FIG. 3a. In particular, the display screen provides the user of electronic terminal 14 with several options. For example, the display screen may provide an area for accessing the Internet from a home location as indicated by area 32 on the display screen, or for accessing the Internet from a remote, travel location, as indicated by area 34 on the display screen. The present invention may also provide to the user the option to access a "first time ?" interface, whereupon the user will be taken to a page providing various comments and instructions relating to first time use. For example, as illustrated in FIG. 3b, if the user

5

clicks on an area associated with "first time" use, the application may provide an individual with detailed instructions regarding the various choices available to the user. Additional information, such as indications that the user will be charged to his or her local phone bill on a per-time basis for Internet access, along with prompts requiring user entry to indicate whether or not the user wishes to continue may be included.

Additionally, the application prompts the user to input various information, such as the telephone number from which dialing is taking place, whether call waiting is a feature (so that it may be disabled, if necessary), and other dialing information (e.g., 9 or 8 access, area code information, etc.). Alternatively, some or all of the input information may be captured automatically, such as with a caller-id feature, etc. Additionally, all gathered data is stored in memory by the application for retrieval upon subsequent uses.

In short, regardless of where the information is provided (e.g., on the opening page, or after accessing one or more pages associated with a "help" feature, or a "first time" feature) when the user initially utilizes system 10, and particularly the software application thereof, the user is prompted to proceed via the "home location" area 32. As indicated at step 38 of FIG. 2, when the user selects Internet access from the home location, processing advances to step 42 for initiating a communications link with access station 12. Alternatively, as will be described in greater detail below, when the user desires to access the Internet from a non-home location, such as is the case when a user has a portable or laptop computer, and, for instance, is traveling and is thus attempting to access the Internet through a telephone line other than the user's home telephone line, the processing advances to step 40, wherein the user is prompted to enter additionally required information. In such an instance, for example, the user may be prompted to input additional information, such as the type of communications line relied upon at the remote location (e.g., pay phone, air phone, hotel, work, someone else's home, etc.). Additionally, information pertaining to the telephone number from which access is being desired may also be required, such as specific dialing instructions necessary from dialing at a remote location (e.g., the requirement to dial a "8" prior to dialing from a hotel, etc.). In either event, electronic terminal 14 may initiate a communications link by dialing a telephone number of the access station (e.g., a toll free or non-toll free number, including but not limited to a 1-800 or 1-900 number), or by otherwise linking with a server associated with an electronic address of the access station.

Once the electronic terminal 14 establishes a communications link with the access station 12, the access station 12, utilizing a processor and software, executes a number of processing steps and makes a number of determinations. In particular, upon the initial time the electronic terminal 14 links with the access station, the access station performs a number of validation functions in an effort to establish an account associated with the electronic terminal 14. As will become apparent from the detailed discussion below, upon subsequent links with access station 12, various validation features are performed, but the account need not be reestablished.

In particular, once a communications link is established between the electronic terminal 14 and the access station 12, an automatic number identification (ANI) process occurs, in which the access station 12 determines the telephone number from which the electronic terminal 14 is communicating. As indicated at step 44 of FIG. 2, access station 12 deter-

6

mines whether the telephone number from which the electronic terminal 14 is dialing has been captured. As indicated at step 46, in the event the telephone number has not been captured, the access station 12 transmits a message back to the electronic terminal 14 prompting the user to action such as call customer service. Alternatively, once the telephone number from which electronic terminal 14 is dialing has been captured and stored at the access station 12, the access station 12 determines whether an account has already been established with respect to the captured telephone number. As will be described below, in the process of initially establishing an account, the access station initially establishes the captured telephone number as a billing telephone number to which charges associated with Internet access time will be billed. As additionally described below, the access station also assigns a user identification and password to the billing telephone number, thereby establishing the account. Accordingly, at step 48, where it is determined whether an account has already been established with respect to the captured telephone number, the processor at access station 12 searches an associated database which stores account information to determine whether the captured telephone number has been assigned to an associated billing telephone number and/or whether the telephone number has an associated user identification and/or password.

In the event an account has not already been established with respect to the captured telephone number, such as is the case when electronic terminal 14 is initially utilized to access the Internet via the system 10 of the present invention, processing advances to step 50. At step 50, the access station 12 determines whether the captured telephone number (e.g., "ANI") is in a particular network that is not affiliated with the access station. In other words, there may be various communications networks with which billing arrangements have not been established, and a determination is made at step 50 whether the captured telephone number is within a predetermined network. In particular, as illustrated, the determination is made whether the captured telephone number is within "OFFNET", which is intended to identify an off network (e.g., one with which a billing arrangement has not been established). It will be understood and appreciated that the determination made may be whether the captured telephone number is within a non-acceptable network or, alternatively, by making a determination whether the captured telephone number is within an acceptable network. As illustrated, in the event the captured telephone number is in a non-acceptable network, such as the case when the captured telephone number is in "OFFNET", processing advances to step 52, wherein the access station 12 sends a message back to the electronic terminal 14 indicating that the telephone number from which access is desired is not valid. Alternatively, when it is determined that the captured telephone number is not within "OFFNET", and therefore may be acceptable, processing advances to step 54, where access station 12 determines the captured telephone number is within a CLEC (competitive local exchange carrier) network database, thus preventing the billing of Internet access charges directly to a telephone bill.

As illustrated, if the captured telephone number is within the CLEC database, processing advances to step 52, and under such circumstances, a message is transmitted from access station 12 to electronic terminal 14 indicating that the telephone number from which Internet access is desired is not valid. However, if the captured telephone number is not within the CLEC database, as determined at step 54, processing advances to step 56, where access station 12 deter-

7

mines whether the captured telephone number has an unacceptable associated credit risk. In this regard, a data base of telephone numbers associated with certain credit risk standards is maintained, such that the captured telephone number is checked against those telephone numbers accumulated in the data base, in an effort to determine whether the captured telephone number is associated with an unacceptable or questionable credit risk. As illustrated, if it is indeed determined that the captured telephone number is associated with an unacceptable or questionable credit risk, processing advances to step 58, wherein the access station transmits a message to the electronic terminal 14 indicating that the telephone number from which access is desired is potentially non-billable, and may further request user action such as a user to telephone the access station 12. However, in the event customer service 12 determines at step 56 that the captured telephone number is not associated with an undesirable or questionable credit risk (e.g., not in a "BADANI" database as illustrated), processing advances to step 60.

At step 60, access station 12 determines whether the captured telephone number is a valid telephone number. In other words, access station 12 checks the telephone number against a line information database (LIDB) in making a validity decision with respect to the captured telephone number. As indicated, in the event a LIDB check results in a determination that the captured telephone number is not valid, processing advances to step 62, and access station 12 sends the electronic terminal 14 a message that the telephone number from which access is desired is not valid, and prompts the user to consider take action such as a call to customer service.

When, however, it is determined at step 60, as a result of the LIDB check, that the captured telephone number is indeed a valid telephone number, processing then advances to step 64, where access station 12 considers the captured telephone number to be a billable telephone number (BTN) (e.g., the number to which Internet access charges will be billed), and access station 12 automatically assigns a user identification and password to the billing telephone number to thereby establish an account with respect to the captured telephone number from which electronic terminal 14 is attempting to access the Internet. Access station 12 then stores the BTN, and associated user ID and password in a database. As indicated at step 66, access station 12 also then sends at that time or at scheduled intervals, via a direct communications link with Internet service provider 24, the account information that has been established with respect to electronic terminal 14. In particular, access station 12 sends to ISP 24 at least the user identification and password associated with the established account. As will become apparent from the following discussion, the Internet service provider 24 will subsequently use the retrieved account information to validate an attempt by the electronic terminal 14 to access the Internet via the ISP 24.

Additionally, once an account has been established at step 64, the processing advances to step 68, where the access station 12 retrieves a table of identifiers (e.g., telephone numbers or server addresses, associated with Internet service provider 24). In other words, the retrieved table includes, for example, modem dialing information associated with Internet service provider 24.

Processing then advances to step 70 at which the access station 12 transmits the established account information, such as the billable telephone number (BTN), user identification, password, and at least one telephone number or address associated with the ISP 24, and retrieved from the dialing table, for the purpose of permitting the electronic

8

terminal 14 to establish a communications link with ISP 24. In particular, and in accordance with the principles of the present invention, the identifier or identifiers (e.g., the telephone number or numbers retrieved from the ISP dialing table), are preferably determined based upon a location of the telephone number from which access is being allowed or from the location of the billable telephone number and, additionally, in the case where multiple identifiers are retrieved, they are preferably prioritized in accordance with selected criteria. In particular, local telephone numbers are given priority.

At step 70 once the access station 12 has returned the billable telephone number, user ID, password, and one or more identifiers (e.g., telephone numbers or addresses) of the ISP 24 to the electronic terminal 14, processing advances to step 74. At step 74, the user is presented with information and billing disclosures (which may be retrieved from the electronic terminal 14, but which are preferably transmitted from access station 12). As indicated at 76, the user is prompted to accept or not accept the proposed billing agreement. An example of a display screen associated with step 76 is illustrated in FIG. 3c. In this regard, and in accordance with an aspect of the invention, the information provided by the billing disclosure indicates that the user will be charged to his or her home telephone number (e.g., the telephone number from which electronic terminal 14 accesses station 12), and the user is given rate information, (such as monetary charges per minute). For example, the stated charges may be a number of cents (USA) per minute. Alternatively, the charges could be per second, per six second increments, or in other increments. As indicated at step 78, if the user elects to deny the billing proposal, processing advances to step 78 and the communications link with the access station 12 is terminated. Alternatively, when the user, at step 76, accepts the billing proposal, processing advances to step 80 and then to step 82, where the communications link between the electronic terminal 14 and access station 12 is terminated, and the electronic terminal 14 initiates a communications link with ISP 24 by dialing ISP 24, respectively. As indicated at step 86, once a communications link has been established between electronic terminal 14 and the ISP 24, the processor at ISP 24 makes a determination of whether the electronic terminal 14 desiring to access ISP 24 is valid. In particular, the electronic terminal 14 sends, via the communications link with ISP 24, the account information that the electronic terminal 14 previously received from access station 12. In particular, electronic terminal 14 transmits data indicative of at least the user identification and password, and the billable telephone number may also be transmitted. A processor at the ISP 24 retrieves the account information data and compares it with the data previously transmitted from access station 12 to determine whether the electronic terminal 14 is associated with a "valid" account. When the ISP 24 is unable to read the account information transmitted by electronic terminal 14, or in the situation where the received account information is not found in the data base of accounts deemed to be valid, log onto the ISP 24 is denied. However, as indicated at step 88, once the account associated with electronic terminal 14 is properly validated, ISP 24 opens a welcome screen or home page, from which the user can browse the Internet or access one or more selected features presented to the user in a conventional fashion.

Additionally, ISP 24 monitors a time associated with a communications link between electronic terminal 14 and ISP 24. In this regard, while the time monitored may encompass the entirety of the time associated with a com-

9

munications link, preferably ISP 24 begins monitoring time only after log on validation procedures have been conducted and the user of electronic terminal 14 is actually logged in to ISP 24. While charges for Internet access time preferably begin after log in, any other communications charges may begin at the establishment of the communications link. It should be understood that the access time may alternatively be monitored from a time associated with access to particular content (e.g., news, video, etc.) or that different or increased rates may apply to the access or downloading of selected content.

Preferably, as indicated at step 90, ISP 24 periodically sends information to access station 12 regarding the amount of time electronic terminal 14 accesses ISP 24. It will be understood and appreciated that this on-line time information, transmitted from ISP 24 to access station 12, is preferably transmitted directly from ISP 24 to access station 12, and not via electronic terminal 14. Additionally, it will be understood and appreciated that the on-line time information may be transmitted from ISP 24 to access station 12 at any preferable interval (such as hourly, daily, etc.). Additionally, the intervals may be varied such that the information is not transmitted at the same time each day.

Returning now to step 48 of FIG. 2, when it is determined that the identifier (e.g., telephone number or server address) is already associated with an established account, processing advances to step 92 at which access station 12 makes a determination whether a previous LIDB check (e.g., that check made at step 60) has been made within a predetermined time frame. For example, as illustrated, the access station 12 queries whether a LIDB response related to the BTN is present. As indicated, if a LIDB response is present in the database, processing advances directly to step 68, and the ISP modem dialing table is retrieved, and processing continues in the manner illustrated and previously described. Alternatively, however, when a determination is made at step 92 that a LIDB response is not present in database, processing advances to step 50, and the illustrated and previously described tree following step 50 is executed. In other words, the step or steps associated with steps 50, 54, 56 and/or 60 are processed, thereby providing a validation procedure.

With reference now to FIG. 4, an example of an invoice sent to the user of electronic terminal 14 is illustrated. In particular, the access station 12, upon receipt of on-line time associated with electronic terminal 14, stores data indicative of the on-line time in the established account corresponding to electronic terminal 14. Accordingly, stored in the account associated with electronic terminal 14 is, for each access event, a date of the access event, a time of day of the access event, the time associated with the access event, and accumulative charge associated with the access event. Additionally, other information associated with the place and telephone number called to achieve the access, rate information, etc., may also be stored in the account in conjunction with each access event. Then, at the end of a selected billing cycle, such as a one month billing cycle, data is retrieved and placed on an invoice along with other charges associated with the billable telephone number, such as long distance charges, taxes, etc. Alternatively, the information retrieved from the Internet access account associated with electronic terminal 14 may be transmitted to another billing entity, for consolidation on a printed or electronic invoice.

From the foregoing, it is seen that the present invention is a highly useful system for accessing the Internet via an electronic terminal, where charges associated with Internet

10

access time are billed directly to a bill associated with the electronic terminal, or the telephone line from which the terminal accesses the Internet. Additionally, the system 10 is useful for approving access only to those electronic terminals 14 or individuals which have predetermined criteria. In other words, access station 12 is useful for preventing access where statistical information indicates that the terminal from which access is desired may be associated with an undesirable credit risk or, alternatively, where the terminal from which access is desired is within a particular network such that billing will be cumbersome or possible, or where the telephone line associated with the electronic terminal is otherwise not a valid line.

Additionally, as will be understood and appreciated, in a preferred embodiment of the present invention, a multitude of electronic terminals 14 are utilized to access the Internet via system 10. In other words, proprietors of access station 12 will have relationships with a wide variety of carriers associated with the communications network, such that a wide variety of individuals may utilize an electronic terminal for accessing the ISP via the access station 12, and thereafter be billed on a per time basis, for the Internet access, preferably on an invoice directly associated with a particular carrier or telephone line.

From the foregoing it will be seen that this invention is one well adapted to attain all ends and objects hereinabove set forth together with the other advantages which are obvious and which are inherent to the structure.

It will be understood that certain features and subcombinations are of utility and may be employed without reference to other features and subcombinations. This is contemplated by and is within the scope of the claims.

Since many possible embodiments may be made of the invention without departing from the scope thereof, it is to be understood that all matter herein set forth or shown in the accompanying drawings interpreted as illustrative, and not in a limiting sense.

What is claimed is:

1. A system for accessing an electronic network via a service provider, said system comprising:

- an electronic terminal having an associated identifier;
- an access station which establishes an account and transmits connection information back to said electronic terminal, wherein the account is established based upon an investigation of a local exchange carrier database;
- a first communication link to link said electronic terminal and said access station wherein said identifier associated with said electronic terminal is transmitted to said access station; and
- a second communications link between said electronic terminal and said service provider, said second communications link initiated between said electronic terminal and said service provider based upon said connection information,

wherein said service provider monitors a time of said second communications link and transmits data indicative of said time to said access station.

2. The system as set forth in claim 1, where in said access station associates said time with said account for billing purposes.

3. The system as set forth in claim 2, wherein said identifier of said electronic terminal is a telephone number.

4. The system as set forth in claim 3, wherein said electronic terminal is portable, and when said electronic terminal is used at a remote location from a home port associated with said telephone number, said access station requires entry of additional information.

11

5. The system as set forth in claim 4, wherein said additional information corresponds to said account.

6. The system as set forth in claim 4, wherein said additional information is indicative of dialing instructions at said remote location.

7. The system as set forth in claim 3, wherein a monetary amount corresponding to said time of said second link is charged to said telephone number.

8. The system as set forth in claim 1, wherein said access station associates a task with said identifier of said electronic terminal, wherein a password is transmitted to said electronic terminal via said first communications link and to said electronic network service provider via a third communications link.

9. A system for accessing an electronic network via an electronic network service provider, said system comprising:

an electronic terminal having an associated identifier and configured to access said electronic network;

an access station, configured to link with said electronic terminal, to store data indicative of said identifier, and to provide said electronic terminal with information associated with said electronic network service provider for use by said electronic terminal in linking with said electronic network service provider responsive to an investigation of a local exchange carrier database, wherein, once a link between said electronic terminal and said electronic network service provider is established, said electronic network service provider monitors a number of time units associated with said established link and transmits data indicative of said time units to said access station for use in billing matters.

10. The system as set forth in claim 9, wherein said identifier associated with said electronic terminal is a telephone number associated with a subscriber.

11. The system as set forth in claim 9, wherein said identifier is an electronic address associated with a subscriber.

12. A method of accessing the Internet with an electronic terminal and via an Internet service provider (ISP), said electronic terminal having an associated identifier, said method comprising:

initially linking said electronic terminal with an access station via a first communications link;

establishing an account at said access station corresponding to said electronic terminal responsive to an investigation of a local exchange carrier database;

transmitting from said access station to said electronic terminal ISP connection information pertaining to said ISP;

using said connection information to link said electronic terminal with said ISP via a second communications link;

monitoring time units associated with said second communications link at the ISP;

12

multiplying said time units by a monetary rate to thereby obtain billing data; and

associating said billing data with said established account for billing purposes.

13. The method as set forth in claim 12, wherein establishing said account further comprises:

determining an identifier associated with said electronic terminal; and

using said identifier to establish a billing identifier.

14. The method as set forth in claim 13, wherein establishing an account further comprises:

establishing a password for said account.

15. The method as set forth in claim 14, further comprising:

transmitting said billing identifier, said password, and an identifier associated with said ISP to said electronic terminal as part of said connection information.

16. The method as set forth in claim 15, further comprising:

additionally transmitting said billing identifier and said password to said Internet service provider via a third communications link.

17. The method as set forth in claim 16, when said electronic terminal links with said Internet service provider via said second communications link, the method including transmitting said billing identifier and said password from said electronic terminal to said Internet service provider and comparing said billing identifier and said password transmitted via said second link with said billing identifier and password transmitted via said third link for log on purposes at said ISP.

18. A method of accessing the Internet with an electronic terminal and via an Internet service provider, said electronic terminal having an associated identifier, said method comprising:

initially linking said electronic terminal with an access station via a first communications link;

establishing an account at said access station corresponding to said electronic terminal responsive to an investigation of a local exchange carrier database;

transmitting from said access station to said electronic terminal ISP connection information pertaining to said ISP;

using said connection information to link said electronic terminal with said Internet service provider via a second communications link; and

associating billing data with said established account at said ISP for billing purposes.

19. The method of claim 18, wherein said identifier is a telephone number.

* * * * *

(C) Evidence submitted by Applicant

The following items listed below are hereby entered as evidence submitted by Applicant.

- (1) Copy UNDERWOOD, Ryan, *To Connect and Serve, Silicon Valley upstart Embrace Networks has unveiled a system that Internet-enables any device, eliminating the need for costly 'one-off' solutions.* Internet On-line April 24, 2001, XP002206279 (1 p.)
<[http://www.embracenetworks.com/news/news 2001-04-24 .html](http://www.embracenetworks.com/news/news%2001-04-24.html)> This evidence was entered into the record by the Applicants on 09/06/2002.
- (2) Copy DUNKEL, Brian et al ., *Customized Metadata for Internet Information*, ISBN 0-7803-3755 - 7/97 1997 IEEE- pages 508-516, (9 pp .) This evidence was entered into the record by the Applicants on 09/06/2002.
- (3) Copy US Patent 6542967 B1 (Major). This evidence was entered into the record by the Applicants on 04/12/2004.
- (4) Copy of Exhibit A, Webster 1828 definition of "provider". This evidence was entered into the record by the Applicants on 10/10/2006.
- (5) Copy of Exhibit B, Webster 1828 definition of "aggregator". This evidence was entered into the record by the Applicants on 10/10/2006.
- (6) Copy of Exhibit A, Webster 1828 definition of "provider". This evidence was entered into the record by the Applicants on 03/17/2008.
- (7) Copy of Exhibit B, Webster 1828 definition of "aggregator". This evidence was entered into the record by the Applicants on 03/17/2008.
- (8) Copy of Exhibit A, encarta® definition of "provider". This evidence was entered into the record by the Applicants on 03/19/2009.

- (9) Copy of Exhibit B, encarta® definition of “aggregator”. This evidence was entered into the record by the Applicants on 03/19/2009.
- (10) Copy of definition of “gateway” and “gateway (telecommunications)” from Wikipedia. This evidence was entered into the record by the Applicants on 08/03/2010.
- (11) Copy of definition of “gateway” from Merriam-Webster. This evidence was entered into the record by the Applicants on 08/03/2010.
- (12) Copy of gateway server manual from Intermec Technologies Corporation for Model 6950 Enterprise. This evidence was entered into the record by the Applicants on 08/03/2010.
- (13) Copy of gateway communication server manual from Klinkmann Automation for MDLC. NB Rev 1.9 only had installation from CD information added thus date of content is Rev 1.8 Apr 200. This evidence was entered into the record by the Applicants on 08/03/2010.
- (14) Incorporated herein by Reference is the entire image file wrapper (Record). N.B. the Record has not been attached due to length and easy availability by the Board. Should this not be acceptable, Applicant reserves the right to attach the entire Record.

Copies of all References follow (except (13)).

//

T Connect and Serve

XP-002206279

Silicon Valley upstart Embrace Networks has unveiled a system that Internet-enables any device, eliminating the need for costly 'one-off' solutions.

By Ryan Underwood

Publish

April 24, 2001

At the height of the Internet frenzy, dedicated evangelists promised that the new technology would change everything about how we live, from the way we shop to the way we receive medical care.

Part of the reason so many believed the Internet might alter the fundamental pattern of human behavior laid in speculation about embedded connectivity. That is, everything from our cars, to our phones, to our bathroom scales would exchange real-time data with a mechanic, a vending machine or a doctor.

But so far that dream has remained mostly just that—a dream. It's not that the technology does not exist. Indeed, embedded Internet technology is even used with great success in some cases. The problem is that there's no standard mechanism for enabling Internet-embedded technologies for a wide range of devices, resulting in high-cost one-off systems that can only be used with a narrow range of products.

Now a company called Embrace Networks has come along with a technology that it claims can deliver on the promise of enabling any device with Internet-embedded connectivity, handling all the details to consummate the needed "handshake" between a device and a service provider.

Len LuPriore, Embrace Network's vice president of marketing, says the technology makes the back-end Internet connection on any device equipped with the company's products and leaves exposed an easy-to-configure XML application programming interface for the device's front end. "We're trying to make sure we minimize the programmatic details of connecting to the Internet," LuPriore says.

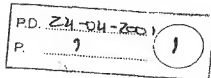
Take the example of an Internet-enabled digital camera. Right now, you take the pictures, upload them to your computer, then upload them to a site where you can create albums, order prints and so forth. With an Internet-enabled camera, you could skip the double upload and just send the pictures straight to whatever Web site you wanted them to go. But, creating a one-off Internet-embedded system for, say a Nikon camera only, is not likely to produce a healthy return on investment. But with a mass-produced system that can eliminate the custom work, the potential ROI starts to look a lot better.

"We kind of look at the Internet as a power grid, where you can get to the Internet from any device" LuPriore says. "We want there to be no need for manual intervention."

Of course LuPriore points out that while Internet-enabled digital cameras would be nice, there are many other areas besides consumer electronics where the technology could be applied. He says doctors could receive heart monitoring or glucose data from patients without having to schedule an office visit, or energy concerns like an Independent System Operator could power down non-essential appliances during idle times to conserve electricity.

The Embrace Network system combines two elements. The first is the client-side hardware and software that controls access between the device and the network. The second piece consists of the server application that controls communication between the network and a service provider. With these two pieces, the company must target device manufacturers as well as service providers as customers. While in some cases they might be one in the same, this isn't always the case.

Founded in 1999 and headquartered in Sunnyvale, Calif., the company says it's raised \$26.5 million from a range of "top tier" VCs and strategic partners. Embrace plans to release its system in the last quarter of 2001 and LuPriore says trials are ongoing in a number of different environments. Asked about pricing, LuPriore could not offer specifics, saying only that the model will be license-based.





RECEIVED

SEP 13 2002

Technology Center 2100

Customized Metadata for Internet Information

Brian Dunkel
Nandit Soparkar
Peter Weinstein

Electrical Engineering & Computer Science
The University of Michigan
1301 Beal Avenue, EECS Building
Ann Arbor, MI 48109-2122
bedunkel, soparkar, peterw@eeecs.umich.edu

Keywords: Customized Metadata, Internet Search

Abstract

Several search engines, catalogs, and filtering services aim to help users of the Internet deal with a growing information "overload". However, these tools typically are either generic in scope, or limited to the needs of a particular user without regard for reuse in some related context. We propose an approach and architecture for customized filtering and cataloging which bridges these two extremes. We allow users to create and maintain a metadatabase of information gathered over time by using modular filters created for specific needs or drawn from a standard library. This metadatabase, which may be regarded as a database view of the Internet, can then be accessed to locate information relevant to specific or more generic tasks. Potentially, our approach achieves greater flexibility and specificity as compared to currently available tools. We describe our preliminary design, implementation, and experimentation for our proof-of-concept prototypical effort.

1. Introduction

The Internet (or the "Web") is already vast, and it is growing rapidly. On any given topic, there

are numerous information sources produced from diverse perspectives. This vast and unstructured information repository leads to an information "overload": Even users with well-defined and ongoing needs, including organizations and professionals, find the available catalogs and search engines too generic to suit their needs. Significant interactive searching and processing is required to obtain the desired information. Compounding the problem is the fact that users often have only a vague or partially defined need for this information.

As an example, consider a human resources department which intends to locate individuals to fill various open employment positions in an organization. Just the fact that many individuals post their resumes on the Web does not provide a list of relevant resumes. A general-purpose Web search engine would prove time-consuming and tedious in terms of obtaining and sifting through irrelevant data. On the other hand, creating a specialized search engine for each such application may prove ineffective in terms of cost. That is, the two available extremes are both inadequate.

Figure 1 identifies three important limitations in current Web tools with respect to their accuracy, flexibility, and the lack of cumulative improvement of search even in a well-defined area.

1. Available search engines support simple queries on keywords or boolean combinations thereof. With keyword-based searching, the results may be inaccurate (i.e., irrelevant information may be found, or relevant entities may be missed). For example, without a proper context there are several connotations associated with the keyword "bridge" (e.g., one relevant to dentistry, another related to civil engineering, etc.).
2. the manner in which synonyms or associated keywords are obtained in search engines is another limitation. They use catalogs which rely upon a fixed, inflexible hierarchy of subjects into which the information is placed. Therefore, they are not as particularly useful for information that represents the intersection of several topics, or do not fit well into the subject classifications (e.g., see [SE95]).
3. In cases where information is relevant over a period of time, generic indexes are modified only with difficulty. Therefore, for specialized needs, or for users with more long term information requirements, there are few available options.

In contrast, our *Customized Meta-Data (CMD)* approach, potentially, could provide the accuracy, flexibility, and opportunity for cumulative refinement required by organizations and professionals with specialized and ongoing needs. Our strategy is to provide a system which allows for re-use of generic components, while giving users the flexibility to configure the information gathering process to suit their ongoing needs. Users would build an information filter to meet their specialized requirements either by selecting from a library of tests and extraction functions created by experts, or by using their own proprietary methods for filtering. CMD would automatically create, populate, and maintain a metadatabase of information as extracted by the filter. Using this metadatabase, more complex and specific queries (e.g., as in database access) can be posed than possible with a generic search engine. In the context of our example above, it would become feasible to search an appropriate metadatabase for resumes of people with at least two years of experience in computer programming (assuming that the appropriate extraction functions had been created).

Figure 2 provides an overview of our approach for CMD. Each element in the figure corresponds to a system component. First, CMD uses some strategy to gather Web information, loading one document at a time. Second, each document is evaluated by a filter, which is a composition of certain tests and data extractors. The tests determine whether a document is relevant, and for those documents which pass the tests, a set of extraction functions returns values (i.e., the metadata) for one or more attributes to be loaded. A commercial version of CMD would have a sizable and sophisticated library of filter tests and extractors, available for flexible configuration by users. Third, a metadatabase maintained using some available commercial data management system would be used to store the extracted keywords.

The remainder of our paper is organized as follows. Section 2 describes a software architecture together with a high-level description of our technique. Thereafter, 3, 4, and 5 describe our preliminary implementation of a CMD approach — the metadata gathering, the filtering, and the metadatabase. We provide a simple motivating example and related discussions in Section 6. Thereafter, Section 7 reviews related efforts and contrasts them with our own. Finally, Section 8 presents our conclusions.

2. System Architecture

Our CMD architecture is divided into four components: the document input, the filters, the database, and the querying interface. Figure 3 illustrates the flow of information between the components, and Figure 4 describes the partitioning of functionality between the components.

A user configures a CMD filter by selecting from lists of test and extractor functions. Tests identify documents to be selected, whereas extractors produce the metadata to be stored in the database. The document input component generates a stream of Web documents, each of which is evaluated by the filters. Metadata extracted from the selected documents is stored in a relational database.

Note that reusability and extensibility are desired objectives motivating the design of the CMD filters. Tests and extractors may be reused in filters targeting diverse types of documents. We intend to add tests and extractors to the available library without requiring modifications to the system itself (beyond the coding of the new functions). Therefore, the domain-specific tests and extractors are encapsulated,

and the interactions are standardized with a well-defined interfaces.

As compared to other information management tools, CMD has some unique characteristics. First, filters may be composed of test and extraction functions that are quite complex. The underlying strategies may range from simple keyword matching or *ad hoc* context-specific rules, to more involved natural language processing. Second, values extracted by the filter are not just keywords found in the text; they could be any relevant descriptive data. For instance, an extraction function operating on resumes might return YEARS_OF_EXPERIENCE although this information is not stated in a document explicitly. In this case, the information may be obtained from the difference between certain start and end dates. That is, derived data (as in a database view) is possible. Third, the metadata base may be refined and improved by CMD users. Therefore, the utility of a metadata base may grow over time as it evolves.

3. Information Gathering

In this section, we describe the initial step leading to the generation of customized metadata for the Internet. We use an example of searching for engineering-related resumes of students made available on the Internet in our local University environment.

Figure 5 shows CMD's search control interface, with options ranging from filtering a predefined list of documents to unrestricted Web crawling. Input documents for CDM may come from a number of sources, with differing degrees of structure. Clearly, when more is known about the structure of an information source, the gathering is more directed and efficient. Often, organizations develop conventions that add domain-specific structure to their Web sites, which may be exploited to increase the efficiency of gathering information.

As an example, using our knowledge of the structure of the Computer Aided Engineering Network (CAEN) at University of Michigan, we began with a list of homepages in the CAEN file system. The HTML were followed links within each homepage and subsequent documents, recursively exploring documents not already visited within the homepage directory and subdirectories. In other words, our hybrid search control (combining link and file crawling) exploits our knowledge of domain-specific structure. The list of homepages is extracted from a list of all accounts maintained by CAEN. Homepages are identified as files named in ac-

cordance with CAEN's convention for homepages (i.e., as `Public/html/index.html` in a user's home directory). Files are accessed by reading them directly over CAEN's file system, although only files with a .html extension are considered. The CAEN web server's mapping from the file directory paths to URL's is utilized since files are read with directory paths, but the URL is included in the metadata.

For less structured domains, or those in which the structure is not known, other approaches may be used to direct the gathering of input documents. For example, the developing class of information brokers (of which CMD may be considered a customizable member) often provide a more structured presentation of available documents from the Web. Existing search engines and catalogs can also provide some direction to the gathering process for potentially relevant input documents.

4. Information Filtering

Our example application was to find relevant resumes, and it is described in more detail in Section 6. Our focus was on the development of functions to configure a filter for resumes, a search control algorithm, programs to create and populate the database, and simple graphical user interface screens for filter creation and search control. For our example, we aimed to search the CAEN system for documents for resumes, and then, to extract relevant metadata from them. However, note that relatively little effort was invested in building highly accurate filters, since the main idea was to validate the basic architecture. The filters we implemented for this simple CMD system are also described in Section 6.

Figure 6 shows our CMD interface for filter configuration. Users name the filter, and select one or more tests. The tests determine which extractor functions are available. Users then select extractor functions, which determine the metadata base fields that are produced. The internal code of the domain-specific filter functions is permitted to be *ad hoc*.

The Test class wraps test functions that determine whether a document should be extracted into the metadata base. Each test has a list of all extractors which are designed to work within the context of documents selected by the test. Every test has two input arguments, a Document and the text describing the HTML link leading to the document (i.e., the text that is underlined in the browser). The result of the test determines whether the document is to be included in the metadata base. The Extractor class wraps

extractor functions, each of which produces metadata for one or more database attributes. Extractors accept the same inputs as tests (a document and text describing a link), and their output is metadata records for the input document. Finally, tests and extractors are stored in a Library object, which returns lists of filter components to the user interface as required.

The database schema determined by a filter is specified by classifying the database fields produced by extractors. Each field has a type, either "not key," "key," or "1-many." All relations have a many-to-one relationship with the document-level relation, either directly or indirectly through an intermediate relation. Typically, all fields produced by an extractor belong to the same relation. Every metadata record produced by an extractor includes all elements of the key of that relation.

Figure 7 shows a portion of the current system which defines an extractor. The Extractor constructor (i.e., the last statement shown) takes six arguments: a name, a pointer to the domain-specific extractor function, the names of the database fields produced, the lengths of these fields, the classification of each field as "key," "not key," or "1-many," and finally, a list of test functions that define the context within which this extractor may be used.

5. Metadata and Querying

In this section, we discuss the storage and retrieval of the metadata generated by our search control and filters. We use a relational database for ease, and provided a graphical query interface. Among the objectives for the CMD metadatabase, the storage schema was to not restrict filter specification beyond the assumption of a relational structure. This independence allows ongoing changes to the various components, and gives developers the flexibility to use other techniques. For example, a different database system could be substituted without modifying the filters. Also, our approach allows for the transparency in efficient querying of the metadatabase.

The data stored includes filter descriptions (i.e., the tests and extraction functions) as well as the metadata itself. The filters and the extracted results are each described by a text file produced by the search module. An abbreviated "grammar" for the filter specification would be:

```
<DATA_FILE> := <FILTER>
               <DATA>

<FILTER> := <NAME>
```

```
<BASED_UPON>
<REPLACES>
<TEST_LIST>
<EXTRACTOR_LIST>
```

Note that either of the <TEXT_LIST> or the <EXTRACTOR_LIST> may be empty; in principle, no assumptions are made about their content.

Once the metadata is stored in the database, it may be queried. Figure 8 shows an example selection from some data collected using the Oracle SQL*Plus query mechanism. More serious CMD mechanisms could include enhanced query capabilities to allow users to retrieve the associated web documents as well by invoking a browser.

6. Generating Specific Metadata

We describe our CMD implementation with regard to the example involving resumes, and the related issues that arise. The filter and search components were implemented in C++ with the Gnu object-oriented library for string handling. C++ was selected for its support for complex data structures and fast execution. The user interface used Tcl/Tk, and the database was coded in Pro*C for an Oracle relational database.

Our CMD search control method began with a list of homepages on the CAEN system, and followed HTML links within the homepage directories. The selected filter functions determined the database schema. The metadatabase included three tables: DOCUMENTS, DEGREES and JOBS. The DEGREES and JOBS were in a many-to-one relationship with DOCUMENTS.

As depicted in Figure 9, after searching approximately 2,000 engineering user homepages (and 10,000 documents in all), CMD selected 186 of them as resumes based on our IsResume filter. Each resume was scanned, and the metadata was written to an intermediate file. From this file, CMD placed the metadata automatically into the three tables. Figure 10 shows the layout of the tables. For each resume selected, there is one record in the DOCUMENTS table. For each degree and job found (since there could be more than one of each per resume), a corresponding record was placed in the appropriate table.

Not surprisingly, the resulting metadatabase indicated that using simple filter functions produces "noisy" data. This is reflected by the fact that 158 of 240 degrees and 426 of 554 jobs

were rejected by CMD due to key conflicts. Also, by reading through the data, a human would note that a few are inappropriate. For instance, a name may appear in the JOB_TITLE field. Obvious reasons for this problem include the simplistic nature of our filter, the lack of sophisticated facilities such as natural language processing, and the heterogeneity of the raw Web data. However, it is a price paid for automating the otherwise tedious manual searches. In order to produce more accurate metadata, more complex filters are needed — and these are accommodated easily within our CMD approach. Also, *ad hoc* filtering could be effected with relative ease.

It would be useful to allow “tuning” of filter functions — for example, by revising synonym lists used for identifying anticipated terms. Another important issue is handling the dependencies between tests and extractors and defining contexts in which they are valid. Currently such dependencies are handled with a structure for annotating documents as they are interpreted, and valid contexts are given by a list of prerequisites for every test and extractor. A major constraining factor for search control on the Web is the size and lack of physical organization. A search initiated at some point may have unbounded traversals through the Web, and therefore, may be inefficient. Instead, a best-first, iterative deepening search strategy to guarantee reasonable performance (e.g., see [RN95]) may be more desirable. Also, by taking advantage of domain-specific information, CMD is able to work quite efficiently. On the other hand, prepared gathering routines, which can take advantage of organization-specific structures, are difficult to obtain.

A CMD system would require some effort by a user mainly in the configuring of filters. To this end, a greater burden rests upon the proper design and implementation of the filters and data-storage mechanisms. Whereas the independence of storage schema and filter specification is difficult to achieve, use of standard database techniques alleviate the problem. One approach would be to use an object-oriented database to help store the filters and metadata efficiently within each domain.

7. Related Work

The rapid increase of search techniques for the Web render any up-to-date surveys ineffective. However, obtaining relevant information that is specific to certain needs remains an elusive goal. Users typically perform keyword searches with rudimentary logical connectives, and therefore, a human is invariably involved in interac-

tive searching which precludes automated techniques. Also, available means to access relevant information, termed information/resource discovery on the Web, are not well-suited to customization. The different search engines are essentially similar; systems such as Harvest [HSW96], Lycos [Lyc], etc., depend on a network of *providers, gatherers and brokers*. A provider site exports information to the network, a gatherer collects this information and organizes it using indexes, while a broker provides a query interface to this gathered information. Our approach is not dissimilar to the conjoining of the two latter components, though with the major difference that we are not limited to keyword searches alone.

Closely related to CMD, SIFT [YGM94] is an information dissemination service implemented for Netnews. In SIFT, a user subscribes to the service by submitting a profile that describes topics of interest. The user then passively receives new, filtered Netnews articles. SIFT provides means to let a user describe a *relevance* threshold, which is the minimum *score* for a document against the user's profile for it to be delivered. Again, the main use is a keyword-based weightage. Typically, keyword-based search engines sum match scores in order to produce a single ranking for the suitability of all candidate documents.

An interesting approach to extend keyword matching, *latent semantic indexing* (e.g., see [FD92]), is a family of techniques analogous to factor analysis used in the social sciences, or eigenvector decomposition in the physical sciences. The advantage of this approach is that keywords are not required to be in a document: matching is a matter of degree, and not a boolean decision. *Patterns* of similar word usage are recognized rather than exact matches, and that provides greater sensitivity to the overall document context. The disadvantage is that a substantial set of representative documents is required to create the word-document matrix used to identify similarities.

An approach to search information based on semantics is being pursued by the knowledge-base community in artificial intelligence. These efforts involve building *ontologies* (formal structures of word meanings), and mechanisms for the logical translations between concepts and phrases (e.g., see [FDFP95]). Information retrieval systems are being developed to combine various methods for keyword matching, refinement of queries using explicit user rating of previous results, and organizing search results (e.g., see [YLY95]). Complex filters, such as those proposed by us, may employ multiple tests

which have a similar or more complicated approach. Some of our multiple filter tests may be well represented using Bayesian inference networks (e.g., see [BC92]). The more sophisticated search and filter techniques create a view of the Web as represented in the metadata — as is the case for CMD.

Systems for organizing cooperative distributed indexing of specialized metadata bases are at an early stage of development. For instance, Harvest is one such attempt with potential to improve the usability of the Web. However, it faces several obstacles similar to those engendered by data and system heterogeneity in multidatabases. In a different context, the Distributed Hypertext Approach [NS91] proposes to present users with a uniform interface to access heterogeneous information repositories. Although an important issue, our current focus in CMD does not deal with heterogeneity directly.

The physical acquisition of raw Web data to be fed through the filters also merits consideration. In this context, we consider the use of domain-specific knowledge, for an organization to efficiently collect metadata as in [KKS95]. Such systems effect file crawling in the local file system, and accumulate a metadata base of keywords and other information to summarize documents.

8. Conclusions

In order to provide a partial solution for searching relevant information within the rapidly growing Web data, we described our customizable and configurable approach to Web indexing. Our approach, Customized Meta-Data, produces domain-specific indexing for the Web information, and thereby produces metadata which may be regarded as a relevant "view" of the Web.

We described a prototype implementation of our approach which consists of a basic gathering process to "crawl" the Web, a configurable filter component to sift through the raw Web data, and the produced customized metadata management and querying. A simple example application experiment was described to illustrate the advantages and disadvantages of our technique.

Using our techniques and tools, we believe that three significant shortcomings of current search engines with regard to providing more focused selections — i.e., accuracy, flexibility, and cumulative improvements — may be substantially mitigated.

References

- [BC92] Nicholas J. Belkin and W. Bruce Croft, "Information filtering and information retrieval: Two sides of the same coin?", *Communications of the ACM*, 35(12):29-38, December 1992.
- [FD92] Peter W. Foltz and Susan T. Dumais, "Personalized information delivery: An analysis of information filtering methods", *Communications of the ACM*, 35(12):51-60, December 1992.
- [FDFP95] Adam Farquhar, Angela Dappert, Richard Fikes, and Wanda Pratt, "Integrating Information Sources Using Context Logic". In *Spring Symposium on Information Gathering from Distributed, Heterogeneous Environments*, 1995. <http://www.isi.edu/sims/knoblock/ss95/>.
- [HSW96] Darren R. Hardy, Michael F. Schwartz, and Duane Wessels, "Harvest: Effective use of internet information — user's manual". Technical Report CU-CS-743-94, University of Colorado at Boulder, 1996. <http://harvest.cs.colorado.edu/>.
- [KKS95] Indu Khosla, Brian Kuhn, and Nandit Soparkar, "Database support for internet information". In *Second International Conference on Applications of Databases*, December 1995. <http://www-personal.engin.umich.edu/~bgkuhn/paper/index.html>.
- [Lyc] Lycos. <http://www.lycos.com/>.
- [NS91] J. Noll and W. Scacchi, "Integrating diverse information repositories: A distributed hypertext approach". *IEEE Computer*, December 1991.
- [RN95] Stuart J. Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, 1995.
- [SE95] Erik Selberg and Oren Etzioni, "Multi-service search and comparison using the metacrawler". In *World Wide Web Journal, 4th International WWW Conference Proceedings*, December 1995. <http://www.w3.org/pub/Conferences/WWW4/Tuesday.html>.
- [YGM94] T.W. Yan and H. Garcia-Molina, "Sift—a tool for wide-area information dissemination". Technical report, Department of Computer Science, Stanford University, 1994.
- [YLYL95] Budi Yuwono, Savio L. Lam, Jerry H. Ying, and Dik L. Lee, "A world wide web resource discovery system". In *World Wide Web Journal, 4th International WWW Conference Proceedings*, December 1995. <http://www.w3.org/pub/Conferences/WWW4/Tuesday.html>.

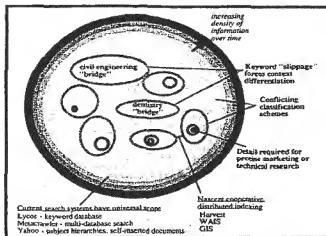


Figure 1: Problems for non-customized Internet indexes

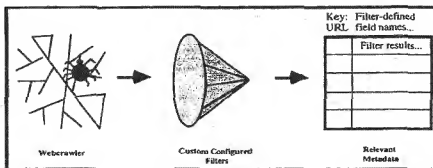


Figure 2: Overview of customizing Internet indexes

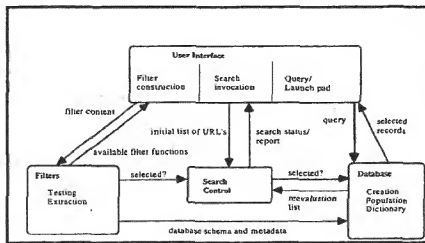


Figure 3: System architecture of CMD

Component	Function
Search Control	Strategies for crawling the web.
Filters	Tests and extractors for metadata.
Database	Creating and maintaining metadatabase.
User Interface	Communicating with users.

Figure 4: Components description

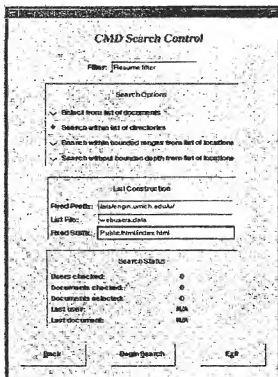


Figure 5: CMD search control interface

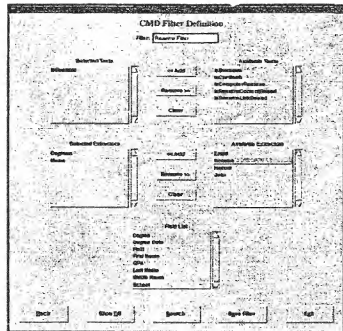


Figure 6: CMD filter definition interface

```
// pointer to the content-specific extractor function
int Degrees(String& linkdesc, Document* doc, char* metadata);

// describes the extractor and the database fields it produces
static char* resume_tests[] = { "IsResume", "IsResumeLinkBased",
                                "IsResumeContentBased", "" };
static char* db_fields1[] = { "DEGREE", "SCHOOL", "FIELD",
                              "DATE", "GPA", "" };
static int field_lengths1[] = { 10, 40, 40, 20, 40, 0 };
static int key_status1[] = { OneToMany, IsKey, IsKey,
                            NotKey, NotKey, 0 };

// construct the Extractor object
all_extractors[i] = new Extractor("Degrees", Degrees,
                                   db_fields1, field_lengths1, key_status1,
```

Figure 7: Definition of the *Degrees* extractor

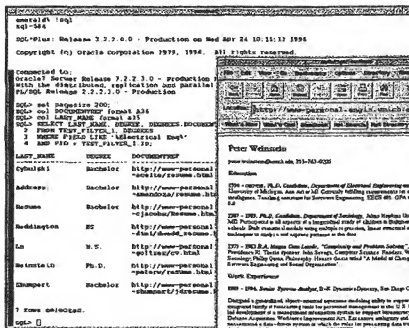


Figure 8: A sample query of the CMD metadatabase

Final report --	
Users checked:	1974
Documents checked:	10173
Documents selected:	186
Last user:	z/s/zxy
Last document:	/afs/engin.umich.edu/a/z/s/zxy/Public/html/index.html

Figure 9: Final status report of a targeted search

SQL> DESCR DOCUMENTS;			
Name	Null?	Type	
ID		NUMBER	
DOCUMENTTYPE		VARCHAR2(256)	
LAST_NAME		VARCHAR2(40)	
FIRST_NAME		VARCHAR2(40)	
MIDDLE_NAME		VARCHAR2(40)	
EMAIL		VARCHAR2(128)	
SQL> DESCR DEGREES;			
Name	Null?	Type	
ID		NUMBER	
DEGREE		VARCHAR2(256)	
SCHOOL		VARCHAR2(40)	
FIELD		VARCHAR2(40)	
GRAD_DATE		VARCHAR2(20)	
GPA		VARCHAR2(40)	
SQL> DESCR JOBS;			
Name	Null?	Type	
ID		NUMBER	
DOCUMENTTYPE		VARCHAR2(256)	
JOB_TITLE		VARCHAR2(40)	
FROM_DATE		VARCHAR2(20)	
TO_DATE		VARCHAR2(20)	
ORGANIZATION		VARCHAR2(40)	

Figure 10: Tables created by CMD for the Resume filter



US006542967B1

**(12) United States Patent
Major****(10) Patent No.: US 6,542,967 B1
(45) Date of Patent: Apr. 1, 2003****(54) CACHE OBJECT STORE****(75) Inventor: Robert Drew Major, Orem, UT (US)****(73) Assignee: Novell, Inc., Provo, UT (US)****(*) Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.**(21) Appl. No.: 09/337,241****(22) Filed: Jun. 22, 1999****Related U.S. Application Data****(60)** Provisional application No. 60/128,829, filed on Apr. 12, 1999.**(51) Int. Cl.⁷ G06F 12/02****(52) U.S. Cl. 711/134; 711/135; 711/113;
711/4****(58) Field of Search 711/160, 161,
711/133, 135, 136, 116, 113, 118, 112,
122, 4, 206, 128; 707/103, 104; 706/16;
710/30****(56) References Cited****U.S. PATENT DOCUMENTS**

5,420,983 A * 5/1995 Noya et al. 701/30
 5,692,129 A 11/1997 Sonderegger et al. ... 395/200,11
 5,717,884 A * 2/1998 Gzym et al. 711/206
 5,737,518 A 4/1998 Grover et al. 395/183,14
 5,761,499 A 6/1998 Sonderegger 395/610
 5,787,439 A 7/1998 Taysom 707/103
 5,794,232 A 8/1998 Mahlum et al. 707/3
 5,832,274 A 11/1998 Cutler et al. 395/712
 5,832,487 A 11/1998 Olds et al. 707/10
 5,859,978 A 1/2000 Sonderegger et al. ... 395/200,56
 5,878,415 A 3/1999 Olds 707/9
 5,903,720 A 5/1999 Stokes 395/186
 5,903,755 A 5/1999 Matheson et al. 395/683
 5,913,025 A 6/1999 Higley et al. 713/201
 5,915,253 A 6/1999 Christiansen 707/103
 5,991,810 A 11/1999 Shapiro et al. 709/229
 6,014,667 A 1/2000 Jenkins et al. 707/10
 6,029,175 A * 2/2000 Chow et al. 707/104
 6,047,358 A * 4/2000 Jacobs 711/128

6,081,900 A 6/2000 Subramaniam et al. 713/201
 6,112,228 A 8/2000 Earl et al. 709/205
 6,128,623 A * 10/2000 Mattis et al. 707/103
 RE36,989 E * 12/2000 White 711/118
 6,157,925 A 12/2000 Jenkins et al. 707/10
 6,163,773 A * 12/2000 Kisshi 706/16
 6,185,612 B1 2/2001 Jensen et al. 709/223
 6,219,676 B1 4/2001 Reiner 707/201

FOREIGN PATENT DOCUMENTS**WO 99/53422 * 10/1999****OTHER PUBLICATIONS**

Kumar et al., "Exploiting Spatial Locality in Data Caches using Spatial Footprints", © 1998 The 25th Annual International Symposium on Computer Architecture, p. 1-12.*
 U.S. patent application Ser. No. 09/195,982, pending, Date Unknown.

W3C Jigsaw, "Setting up Jigsaw as a Proxy", <http://www.12.w3.org/Jigsaw/Doc/User/proxy.html>, Jun., 1998, pp. 1-5.

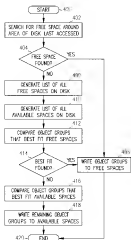
Sun Microsystems, Inc., "Java Servlet API", <http://www.java.sun.com/products/servlet/index.html>, Copyright 1995-1998, pp. 1 and 2.

Sun Microsystems, Inc., "Java Server", <http://www.javasoft.com/marketing/enterprise/javaserver/html>, Copyright 1995-1998, pp. 1 and 2.

* cited by examiner

Primary Examiner—Do Hyun Yoo**Assistant Examiner—B. R. Peugh****(74) Attorney, Agent, or Firm—Schwegman, Lundberg, Woessner & Kluth, P.A.****(57) ABSTRACT**

A cache object store is organized to provide fast and efficient storage of data as cache objects organized into cache object groups. The cache object store preferably embodies a multi-level hierarchical storage architecture comprising a primary memory-level cache store and, optionally, a secondary disk-level cache store, each of which is configured to optimize access to the cache object groups. These levels of the cache object store further exploit persistent and non-persistent storage characteristics of the inventive architecture.

22 Claims, 4 Drawing Sheets

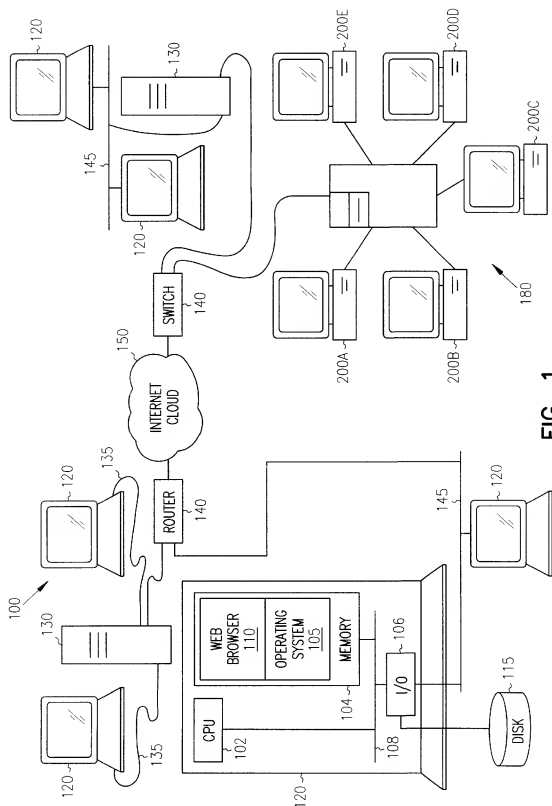


FIG. 1

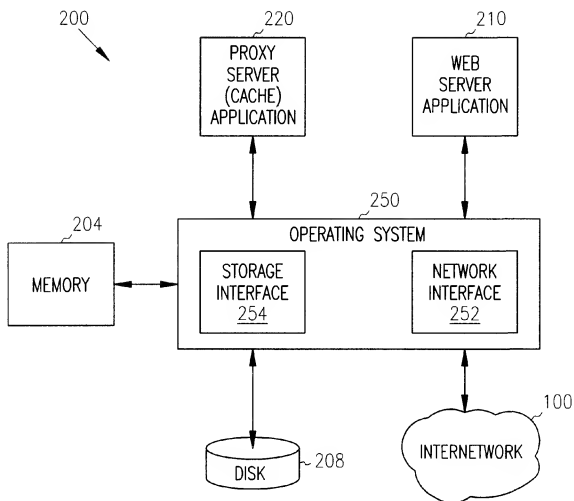


FIG. 2

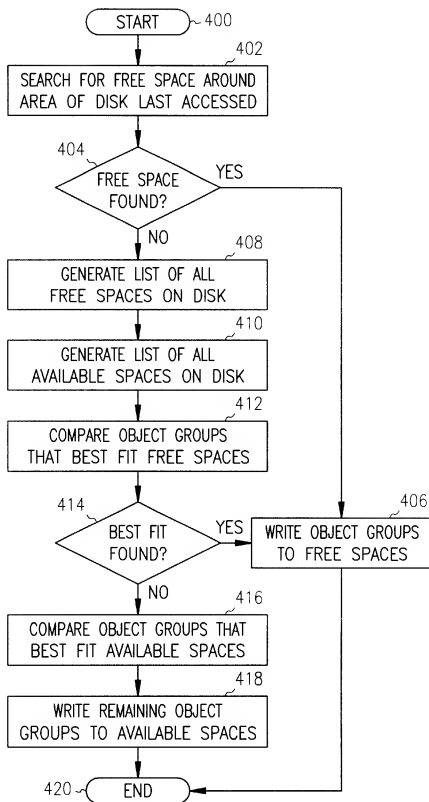


FIG. 4

CACHE OBJECT STORE

REFERENCE TO PROVISIONAL APPLICATION

This patent application claims the benefit under 35 U.S.C. §119(e) of U.S. Provisional Patent Application No. 60/128,829, now U.S. Pat. No. 6,112,228, titled, Cache Object Store, by Drew Major and filed on Apr. 12, 1999, which application is hereby incorporated by reference as though fully set forth herein.

CROSS-REFERENCE TO RELATED APPLICATIONS

This invention is related to the following copending and commonly assigned U.S. Patent Applications:

U.S. patent application Ser. No. 09/023,895, now U.S. Pat. No. 6,112,228, titled, Client Inherited Functionality Derived from a Proxy Topology where each Proxy is Independently Configured by Douglas G. Earl et al, filed on Feb. 13, 1998; and

U.S. patent application Ser. No. 09/195,982, now U.S. Pat. No. 6,330,605, titled, Proxy Cache Cluster by Brent Ray Christensen et al, filed on Nov. 19, 1998.

BACKGROUND OF THE INVENTION

Users having computers interconnected by an institutional intranet or local area network may access various remote sites (such as those on the "World-Wide Web") via the well-known Internet communications network. Using resident web browser applications executing on the computers, these "clients" may navigate among data ("pages") stored on various servers ("web sites") and may further view the contents of these pages as desired. In a basic network communication arrangement, clients are free to access any remote web site for which uniform resource locator (URL) addresses are available. It is increasingly common in network applications to provide each client with access to a so-called proxy server that links to the Internet. A proxy server accesses requested data from the web sites and stores it locally (i.e., "caches" the data) to effectively speed-up client access and reduce the download time of future requests for the data. In response to a request from a browser executing on a client, the proxy server attempts to fulfill that request from its local cache storage; if it cannot, the proxy server forwards the request over the Internet to a server that can satisfy the request. The server then responds by transferring a stream of data to the proxy server, which caches and forwards the data onto the client.

Caches have become increasingly important in the context of proxy servers as the amount of data consumed over networks increases. A cache system typically stores a subset of an entire data set in its store and that data is constantly rotated in and out of the cache in accordance with an algorithm that identifies the data to be replaced, such as a conventional least recently used algorithm.

The cache system is not the primary source of the data set and, therefore, it can retrieve any data that has been deleted or lost from a source that "publishes" the data. Despite increasing network bandwidth, it is desirable to cache data closer to the consumer is of that data, especially as local client access speeds and content density increase. In this context, closeness is defined in terms of bandwidth or accessibility to the data so as to enhance a user's experience. The typical Internet model wherein the publisher of the data is provided with substantial content capacity, i.e., the ability to service (or deliver) content as requested, is fundamentally

non-scalable. Wide spread use of caching technology increases scalability and decreases content access requests at the content origin site.

Cache systems generally rely on the ability to organize access requests in a fast storage mechanism, such as memory composed of random access memory devices. If the cache of a proxy server is servicing a busy communications channel, it will eventually exhaust the memory. At this point, the system may (in accordance with the conventional replacement algorithm) either discard portions of the cached data or move those portions from memory to another storage mechanism, such as a disk. Although this latter option increases the persistency of the cached data and extends the amount of cache memory, it also introduces a relatively slow storage mechanism into the cache system.

Two common paradigms for the persistent storage of data are file systems and database systems. A file system contains general knowledge of the organization of the data stored on storage devices, such as memories and disks, needed to implement properties/performance of a desired storage architecture. A database system provides as structured data model that may be implemented on a file system or other storage architecture. Notably, there is an expectancy that the data (i.e., "content") stored on the file system or database will be preserved until explicitly removed. Persistency with respect to the storage of content, e.g., naming of data files and their non-volatile storage, is paramount to other properties/performance metrics such as organization of, and speed of access to, the stored content. As such, these characteristics of a file system or database are not generally suited to the access and volatility characteristics of a cache system.

Conventional file systems have evolved to take advantage of the higher disk densities but have not generally overcome limitations of the number of disk operations per second. Disk density/capacity generally increases on a price/performance curve similar to that of semiconductor technologies by, e.g., making disk tracks thinner. However, disk access times are not decreasing at the same rate due primarily to physical constraints; indeed, the number of disk operations per second is increasing only minimally due to rotational latencies and head-throw seek times.

Therefore, a feature of the present invention is to provide a cache system that efficiently retrieves and stores data transferred over a computer network.

Another feature of the present invention is to provide a cache system having features of a persistent store and a non-persistent store.

Yet another feature of the invention is to provide a cache system that includes volatile and non-volatile (e.g., disk) storage capabilities.

Yet another feature of the invention is to provide multiple memory abstractions that allow exploitation of the characteristics of a cache environment.

Still yet another feature of the present invention is to provide a cache system that includes a mechanism for reducing the number of disk operations needed to store data and that advantageously utilizes disk density.

SUMMARY OF THE INVENTION

The invention comprises a cache object store organized to provide fast and efficient storage of data as cache objects, which can be organized into cache object groups. The cache object store preferably embodies a multi-level hierarchical storage architecture comprising (i) a primary memory-level

3

(RAM) cache store and (ii) a secondary disk-level cache store, each of which is configured to optimize access to the cache object-groups. These levels of the cache object store further cooperate to provide an enhanced caching system that exploits persistent and non-persistent storage characteristics of the inventive architecture.

In the illustrative embodiment of the invention, the memory-level and disk-level stores are optimized as fast cache components by exploiting the characteristics/attributes of memory and disk storage devices constituting these stores. For example, the memory devices are configured to be efficiently accessed on "natural" boundaries to conform with address mapping arrangements, whereas the disks are optimized for such attributes as geometry, head movement and sector interleaving. If another tertiary-level cache is used in the hierarchical architecture, those storage devices would be similarly characterized and advantageously employed.

A cache object manager implements various aging and storage management algorithms to manage the cache object store. An example of such an aging policy is a modified least recently used (LRU) algorithm that strives to keep those object groups that are accessed most often in the primary-level cache store, with as many remaining object groups stored on the secondary-level store for quick retrieval. According to the cache object manager policy, each object group is marked with a time of last access that indicates the frequency at which object group is accessed within the cache store, and a cost of reacquisition to determine which object groups to move or delete.

A cache directory manager cooperates with the cache object manager to implement the storage management policies. The secondary-level store is primarily used to locate certain object groups from memory to disk if the aging mechanism recommends relocation. Relocation of an object will result in one of three states: object in RAM only, object in RAM and disk, or object on disk only. The cache directory manager maintains lists of object groups to be moved for each object group size. The storage management policy seeks to optimize movement of cache object groups from memory to disk by, e.g., moving the disk head as little as possible.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and further advantages of the invention may be better understood by referring to the following description in conjunction with the accompanying drawings in which like reference numerals indicate identical or functionally similar elements:

FIG. 1 is a block diagram of a network including a collection of network segments connected to a plurality of client and server computers, the latter of which may be organized as a service provider;

FIG. 2 is a highly schematized diagram of software components of the service provider server of FIG. 1;

FIG. 3 is a schematic block diagram of a multi-level hierarchical architecture of an inventive cache object store comprising a primary-level cache store and a secondary level cache store, each of which is configured to optimize access to cache object groups in accordance with a cache object manager according to the invention; and

FIG. 4 is a flowchart illustrating a sequence of steps followed by a cache directory manager when executing a storage management policy with respect to moving a cache object group from the primary-level cache store to the secondary-level cache store in accordance with the present invention.

4

DETAILED DESCRIPTION OF AN ILLUSTRATIVE EMBODIMENT

FIG. 1 is a schematic block diagram of a computer internetwork 100 comprising a collection of network segments connected to a plurality of computers 120, 130, 140 and 200. Each computer generally comprises a central processing unit (CPU) 102, a memory unit 104 and an input/output (I/O) unit 106 interconnected by a system bus 108. The memory unit 104 may comprise storage locations typically composed of random access memory (RAM) devices, which are addressable by the CPU 102 and I/O unit 106. An operating system 105, portions of which are typically resident in memory and executed by CPU, functionally organizes the computer by, inter alia, invoking network operations in support of application programs executing on the CPU. An example of such an application program is a web browser 110, such as Netscape Navigator™ available from Netscape Communications, Inc.

The I/O unit 106 connects the computer to the network segments and to at least one mass storage device 115. As described herein the mass storage device, such as a disk, may function as a component of an enhanced cache system for storing information relating to, e.g., content available on the network. Typically, the I/O unit 106 provides information, such as control and data signals, to the CPU 102 for storage/retrieval of data from the disk or for transfer over the network segments.

The network segments may comprise local area networks or intranets 145, point-to-point links 135 and an Internet cloud 150. Collectively, the segments are interconnected by intermediate stations 140, such as a network switch or router, and configured to form an internetwork of computers that communicate by exchanging data packets according to a predefined set of protocols, such as the Transmission Control Protocol/Internet Protocol (TCP/IP). It should be noted that other techniques/protocols, such as Internet Packet Exchange (IPX) protocol, File Transfer Protocol (FTP), and the Hypertext Transfer Protocol (HTTP), may be advantageously used with the present invention.

In the illustrative embodiment, the internetwork 100 is organized in accordance with a client/server architecture wherein computers 120 are personal computers or workstations configured as clients for interaction with users and computers 130, 200a-e are configured as servers that perform services as directed by the clients. For example, the servers 200 may be configured to operate as a service provider (e.g., an Internet service provider 180 such as concentric.net) as described further herein, whereas servers 130 may be configured as domain name system (DNS) servers and/or Internet provider servers. In general, the DNS servers provide the clients 120 with the network (e.g., IP) addresses of requested services in response to packets directed to the domain names of those services. The Internet providers, on the other hand, provide Internet access to the clients via, e.g., dial-up telephone lines or cable links.

The client 120 may further utilize the web browser 110 to gain access to the web site 180 and to navigate, view or retrieve services stored on the web servers 200. The pages are typically hypertext documents created using a conventional hypertext markup language (HTML) format. In order to effectively speed-up access to the service provider and reduce the retrieval time for stored services, each web server 200 may be provided with access to a proxy cache server. FIG. 2 is a highly schematized diagram of software components of the web (service provider) server 200. These components generally include an operating system 250

5

having utility programs that interact with various application program components to provide functionality, including a network interface for communication with a client browser 110 over the network 100. The application program components include a web server application 210 and a proxy server application ("proxy") 220.

Since the proxy 220 "front-ends" the web server, the network address of the proxy (rather than the actual web site) is published in the DNS server 130 and that address is mapped to the domain name of the service provider. To access a service of [the service provider] web site 180, the client sends a request packet directed to the network address of a particular proxy 220 of the web site. The proxy 220 receives the request from the browser 110 and, if the client is authorized to access services from the web site, the proxy attempts to fulfill that request locally from information stored, e.g., in memory 204 or on disk 208; in either case, the memory and/or disk cooperate to provide an enhanced cache system for quickly storing and retrieving the services. If it cannot satisfy the request, the proxy forwards the request onto the web server application 210. The web server application then responds by transferring a stream of information to the proxy, which stores (caches) and forwards the information onto the browser 110. Although the proxy 220 is shown as resident on the web server 200, it should be noted that the proxy may also be configured to run on a separate server platform.

Cache Object Store

FIG. 3 is a schematic block diagram of the enhanced cache system which is preferably organized as a novel cache object store 300 to provide fast and efficient storage of data as cache objects 302. A cache object is characterized as a collection of data that is persistent over a predetermined period of time, but that can be recovered if lost. For example, if a user request is made to the proxy 220 to retrieve the contents of a home web page 308 (such as an HTML page) from a web site 180, the HTML page and the contents of all references comprise a cache object group. Likewise if the request is directed to retrieving a file using, e.g., the file transfer protocol (FTP), the cache object is the file to be retrieved. Characterization of a cache object thus generally applies to the form of a cacheable data structure that is responsive to, and therefore defined by, a particular request.

In the case of an HTML page, the content of that page generally references additional uniform resource locators (URLs) 309 and their content. The user that has requested access to the HTML page will likely reference the URLs present on that page; all of the content that is likely to be referenced next may be defined as a cache object group 305. Organizing such information as a group is advantageous because a request to a particular page of a web site would likely be followed by accesses to the URL links present on the page. During a write operation affecting any portion of the cache object group, the entire object group is written. Later during a read operation, the entire group is retrieved to memory. Note that other definitions of a cache object group 305 may be used to address content characteristics.

The cache object store 300 is generally associated with a proxy cache within the internetwork; in the illustrative embodiment, however, the cache object store is associated with a Processor Memory Mechanism (PMM) of a proxy cache cluster (PCC). An example of a PCC system including a PMM that may be advantageously used with the present invention is described in copending and commonly-assigned U.S. patent application Ser. No. 09/195,982, titled Proxy

6

Cache Cluster, by Brent Ray Christensen et al., which application is incorporated by reference as though fully set forth herein.

In accordance with the present invention, the novel cache object store 300 preferably embodies a multi-level hierarchical storage architecture comprising (i) a primary-level cache store 310 and (ii) a secondary-level cache store 320, each of which is configured to optimize access to the cache objects. The hierarchical architecture of the cache object store may be extended to accommodate another tertiary-level store 330 that may be slower than the secondary-level cache. For example, the hierarchical storage architecture may comprise volatile memory (RAM) 204 (or equivalent media) as the primary-level cache, non-volatile disk(s) 208 (or equivalent media) as the secondary-level cache and bubble memory 332 (or slower disk) as a tertiary-level cache. However, it should be noted that the inventive storage architecture may operate solely with a primary-level cache.

The various levels of the cache object store cooperate to provide an enhanced caching system that exploits persistent and non-persistent storage characteristics of the inventive architecture. For example, assume a cache object group 305 that is stored entirely in the primary memory store 310 is overwritten and is subsequently requested. Broadly stated, the cache object store 300 first checks the secondary cache store 320 for the object group and, if it is stored on disk, the object store retrieves the group. However, if the group was not stored on disk, the object store retrieves the content from its source in accordance with the non-persistent nature of a conventional cache.

In the former case described above, the persistence of the secondary cache store enables rapid reloading of the primary cache to a useful serving state according to the algorithms described herein. If a substantial period of time has passed before restoring the previous content state of the primary cache, most of that content is invalidated because of, e.g., specified Time-To-Live (TTL) values, whereas if only seconds or minutes has passed, most of that cache contents would be considered valid and available for serving because the TTL has not expired. In the latter case, the cache object store may be on a different platform than the source server; in that case, the novel object store "goes back" to the actual source server to fill the request and obtain the information again. Alternatively if the information is cached somewhere closer to the cache object store, the request may be directed to that closer cache.

In the illustrative embodiment, the memory-level and disk-level stores are optimized as fast cache components by exploiting the characteristics/attributes of RAM and disk storage devices. For example as described further herein, the RAM devices are efficiently accessed through hardware registers on "natural" segmented boundaries, whereas the disks are optimized for, e.g., geometry, head movement, cylinder versus track and sector interleaving. If another storage device is utilized as a tertiary-level cache, it would be similarly characterized and advantageously exploited.

A cache object manager process 350 controls the cache object store by implementing various aging and memory management algorithms, along with recording the number of data buffers that are allocable. For example, the cache object manager implements aging policies, such as a modified least recently used (LRU) algorithm described further herein. The cache object manager 350 also controls aging with respect to time of access and cost of re-acquisition of the data to thereby manage the full hierarchy (each level) of the cache store by, in part, determining when to remove select cache

object groups. The cache object manager is preferably implemented as a state machine process that is executed, e.g., on the PMM platform; however, it will be apparent to those skilled in the art that the cache object manager may be implemented in hardware and may even reside on a different platform, e.g., in a shared memory model arrangement, as long as it has access to all of the cache memory.

When fetching new cache object (URL) groups from their origin, the proxy 220 examines the HTML pages 308 and determines which other URLs 309 are referenced (this is applicable to any content that has determinable pointers). The cache object manager uses this information to pre-load the cache object store 300 prior to the client requesting those URLs. There is generally only one copy of a URL in the cache object store except when a URL is sent to a consumer and another consumer asks for a refresh and/or the time-to-live (TTL) expires and there is a new copy that needs to be filled from the origin.

Cache object groups 305 are generally defined as references in some markup language or mechanism (e.g., HTML) and can be in one of two states: (1) the original page (e.g., HTML) is cacheable; and (2) the original page (e.g., HTML) that was used to define the object group is itself non-cacheable. This implies that when this page is referenced again it may return a different set of URLs. For example, consider a page with ads that are changed each time the page is viewed; many of the URLs on this page are the same but the added URLs may be different each time. A particular URL may be defined as being in more than one object group, e.g., a logo that appears on several web pages. A cache grouping function 370 logically associates these URL/page objects onto an object group 305 so that the entire group is moved to the cache object store 300. The cache grouper 370 is preferably invoked via Define ObjectGroup call issued by the cache object manager 350. According to the modified LRU aging policy, a predetermined percentage (e.g., 25%) of the last accessed cache object groups 305 are targeted for movement to a lower level in the memory hierarchy or for deletion at any time. The policy strives to keep those objects that are accessed most often in primary-level memory 310, but if they are written to [disk] secondary-level cache store 320, they still can be quickly retrieved. If they are deleted then reacquisition is necessary. If there is free space on the disk and the disk is idle, all object groups in RAM are written to disk. Each object group 305 is marked with a time of last access (TLA). A data structure, such as an aging table 352, is maintained by the object manager 350 that tracks the frequency at which object groups are accessed (TLA factor) within the cache system. The technique also factors in a cost of reacquisition (COA) in determining which object groups to move or delete. For example, cache object groups that originate from close sources and have a "low cost" of reacquisition are marked available for movement or deletion, while those objects that are expensive to reacquire are kept longer.

A strict LRU policy is not used because, if possible, it is desirable to avoid head-seek operations that require head movement as described further herein and targeting the last accessed objects makes it likely that there is free space close to current location of one of the disk's read/write heads. In other words, enforcement of strict LRU order when purging old cache objects may result in excessive head movement throughout the disk. Rather the modified LRU policy is employed that targets some selection of those objects for deletion based on relative computation of the last access times and reacquisition costs. For instance, if an object group is frequently accessed and its cost of reacquisition is

high, that group will generally not be deleted. Instead, an object/group may be deleted that has a lower acquisition cost, despite being accessed frequently.

The cost of reacquisition may be empirically determined by measuring the turn-around time needed to acquire an object/group (of course, other mechanisms may be characterized which determine some cost of reacquisition). The turnaround time is defined as the time elapsed from when a request is issued to acquire an object to when the object is actually retrieved. For example, it may take 8 milliseconds to turnaround a request for 4,000 packets. This may be considered a relatively expensive reacquisition because the acquisition time is long and the size of the object/group is large. The modified LRU algorithm preferably factors in the cost of reacquisition in the following manner. Each object is tagged with a value between, e.g., 0 and 7, thereby providing eight (8) levels of reacquisition cost, wherein 0 is the most expensive and 7 the least expensive. The reacquisition cost is then used to factor the times when objects in each of the categories become available. The times are all relative to each other and may be presented on, e.g., a logarithmic or half scale basis.

For example, if the access time for a reacquisition cost level of 0 is 27 minutes, then the access time for level 1 is 18 minutes, the access time for level 2 is 12 minutes, the access time for level 3 is 8 minutes, the access time for level 4 is 5 minutes, the access time for level 5 is 3 minutes, the access time for level 6 is 2 minutes and the access time for level 7 is 1 minute. Alternatively, if the access time for level 0 is 512 minutes, then the access time for level 1 is 256 minutes, the time for level 2 is 128 minutes, the time for level 3 is 64 minutes, the time for level 4 is 32 minutes, the time for level 5 is 16 minutes, the time for level 6 is 8 minutes and the time for level 7 is 4 minutes. Note that the number of reacquisition levels can be easily changed, as well as the relative time differential between levels. Also note that even though a particular object is capable of being deleted, it may not actually be deleted. Movement or deletion of objects depends on, inter alia, the location of the disk head and the amount of data to be written.

Primary Level Cache

In the illustrative embodiment, the cache object store 300 utilizes all available memory 204 for storing ("caching") frequently accessed cache object groups 305 at the primary level store 310 with as many remaining object groups being cached on the secondary (or lower) level store(s) as possible. Note that a cache object group is preferably stored entirely in memory 204 or entirely not in memory, although this characteristic is not fundamental to the present invention.

The memory-level store is optimized by accessing the RAM devices through e.g., hardware index and displacement registers on a "natural" (4 K or 64 K) boundary, primarily because addressing is commonly mapped to such segmentation. The primary storage mechanism in RAM is a data buffer 312 preferably configured to accommodate a predetermined cache object size. The data buffers are used to "buffer" both data and metadata so that all disk operations occur through these buffers to reduce copies and transformations. Note that there are certain control structures that are not written to disk and thus do not use the data buffers.

In general, the cache object store renders disk write accesses as background operations and reserves all foreground disk capacity to read operations. That is, the cache object store does not write data to disk until it makes sense to write (e.g., the disk is idle and data matches available

space at the current head location). Further, the cache object store **300** does not require conventional directory searches because it often finds the requested data promptly via in-RAM data structures with only one disk operation to retrieve or write the data and metadata.

An example of such an in-RAM data structure is an In-RAM cache object store directory **314**. Each cache object group preferably has a representation within a directory in memory **204** that is used for mapping cache object groups to disk **208**. Rather than marking a cache object group as deleted, the In-RAM directory is modified to indicate that the block of memory occupied by the group is available. The cache object store directory **314** is periodically written to disk (as governed by a predetermined policy); if the system goes "down", the directory is read from disk (on remount) and used to rebuild the In-RAM directory. When rebuilding the directory **314**, bad (overwritten) object groups are detected and invalidated; in this case, the characteristics of the cache object store are used to re-fill the store from the source(s). As object groups are accessed during normal operations bad (overwritten) object groups are detected and invalidated if necessary.

The In-RAM cache object store directory **314** is characterized by a predetermined searching technique to rapidly located the object group in the cache store **300**. It will be apparent to those skilled in the art that there are many ways to implement such an in-RAM directory, such as a tree-based or binary search organizations; however, the preferred embodiment uses hash-based URL searching to reduce the domain with which to manage information. The in-RAM cache object store directory **314** is generally coherent and up-to-date unless a failure occurs; in the event of a failure, it is rebuilt from the On-Disk object store directory when remounted.

A cache directory manager process **360** comprising a memory directory sub-manager **362** and a disk directory submanager **364** are adapted to manage the organizations of their cache object stores to implement efficient storage management policies. By moving the objects to disk, the invention characterizes the objects as "safer" to remove/delete from primary memory with newer, more active objects. The cache directory manager **360** maintains lists **366** of object groups for movement to disk for each object group size. When the objects **302** are completely defined, the object groups **305** are added to these lists. These lists **366** are preferably maintained in a first-in, first-out manner so that changes to the object group definition are likely to be realized before "flushing to disk" a first time. Note that secondary store is not required for correct operation of the invention, i.e., cache can operate entirely out of RAM.

FIG. 4 is a flowchart illustrating the sequence of steps followed by the cache directory manager when executing a storage management policy with respect to moving a cache object group **305** from primary to secondary cache store **320**. In general, the cache object store **300** and, in particular, the disk directory submanager **364** seek to optimize movement of cache object groups from memory to disk by, e.g., avoiding fragmentation of those object groups on disk. The sequence starts in Step **400** and proceeds to Step **402** is where a search is conducted for free space (i.e., over a preconfigured level) around the area of disk last accessed. If free space is discovered (Step **404**), the cache object group is written to that free disk space in Step **406**. If there is no free space found (Step **404**), then a list of all free spaces on the disk is generated in Step **408**. In Step **410**, a list of used, but available spaces on disk is also generated.

In Step **412**, combinations of object groups (l-n) are compared that would best fit one of the listed free spaces.

Preferably, the comparison operation favors larger, completely free spaces, i.e., it is desirable to write the object group to only free space. If combinations of object groups are found that fit the free spaces (Step **414**), then the cache object groups are written to that free disk space in Step **406**. If not, combinations of object groups (l-n) are compared that would best fit one of the listed available spaces in Step **416**. This step involves calculating available space by coalescing (fusing) boundaries between the two lists; again, this comparison operation favors larger free spaces. The remaining cache object groups are then written to the available spaces in Step **418** and the sequence ends at Step **420**. As many object groups as possible will be written per write operation. Large object groups are given preference over small object groups.

Referring again to FIG. 3, the cache object store **300** may be characterized as a modified log structured file system that minimizes the number of disk operations by (i) mixing "metadata", i.e., data pertaining to the file system structure, with the actual data and (ii) orientating itself towards large contiguous write operations. The cache object store is a modified log structured file system because it can lose data stored thereon as a result of, e.g., server or disk failures, whereas a file system guarantees the integrity of stored data despite such failures. That is, the cache object store exploits the fact that it is just a cache and can always retrieve the data later.

Secondary Level Cache

In the illustrative embodiment, the cache object store may use a plurality of disks **208** as the secondary cache store **320**. The cache object manager **350** optimizes the use of secondary storage by characterizing each disk in terms of geometry and its natural formatting using, e.g., cylinder, track and sector interleaving. If sector addressing is not desired, another kind of addressing scheme, such as clustering or "blocking", may be overlaid on sector addressing. Disk space is preferably allocated through disk sectors **326** and a plurality of disk sectors comprises a disk block **328**. For example, a plurality of sectors may be organized as one addressable disk block of, e.g., 1 KB, 4 KB, 16 KB or 256 KB.

Disk blocks **328** can be used in a plurality of ways including (i) as the on-disk object group directory **324** used to rebuild the in-RAM object group directory **314**; (ii) for storing actual data/metadata; (iii) as unused blocks; and (iv) as bad blocks including those blocks with I/O errors that are no longer used. The maximum size of a cache object group **305** is preferably 256 KB, although other object group sizes may be employed. Those objects or object groups that are greater than 256 KB are preferably apportioned into multiple 256 KB blocks.

In accordance with an aspect of the present invention, the cache object store **300** preferably writes ("stores") data in the vicinity of a last read or write operation. For example when storing cache object groups **305** on disk **208**, the cache object store may skip certain disk sectors **326** to avoid erasing data contained therein; in contrast, a log structured file system typically performs a "garbage collection" cycle that moves the data before overwriting the sectors. A log structured file system typically performs a seek operation followed by a write operation, whereas the cache object store often avoids such seeking. Unlike a log structured file system, the cache object store may not update a "master" record (e.g., directory) as part of a write operation.

A goal of the invention is to minimize read/write transducer **322** (head) movement when caching data on the disk

208 because such movement causes substantial delay when accessing the secondary storage level 320. Furthermore, a cache object group 305 is preferably accessed during each read/write operation to optimize disk accesses. Disk accesses preferably occur at locations close to the last access to minimize head movement for write operations while realizing performance increases with read operations. Although this may result in overwriting of cache object groups, the invention exploits the fact that the cache object store is not the original repository of the cached information and that it can retrieve the overwritten data again, if necessary.

As noted, all objects within a cache object group are preferably accessed together. A trailing signature 306 at the end of each object group 305 may be used to validate consistency; if the trailing signature is different than a leading signature 304 of the group, this may indicate that the object or object group was not completely written to disk. The disks 208 retrieve multi-sector data blocks 328 in disk order (i.e., data blocks are not initially retrieved from the middle of the sectors to thereby obviate faulty leading and trailing signature checking).

The cache object store treats each disk 208 independently and each object group 305 is always stored entirely on only one disk. Each cache object store on disk is accessed as a separate partition. Partitioning of the disk enables marking/reserving a portion of each disk as the cache object store, labeling that reserved portion as proprietary and non-accessible by other processes executing on the platform.

As also noted, a complete directory of the cache objects/groups stored in the primary level cache 310 and the secondary level cache 320 is maintained on the In-RAM cache object store directory 314 in memory 204. This allows monitoring of storage access activity among various areas of, e.g., the disk in order to implement storage management policies, such as ensuring utilization of all areas of the disk by prioritizing certain activities and areas for caching data. For example, the cache object store 300 prioritizes disk read accesses by attempting to perform disk write operations only on idle disks. In a system with multiple disks 208, the cache object store performs parallel disk accesses.

The cache object store 300 characterizes each cache object group to determine which group is "flushed" from memory 204 to disk 208 when the cache object store exhausts its memory while servicing a busy channel. Use of the secondary level store 320 increases the persistency of the cache object store because, rather than deleting an "aged" object group, the aging algorithm moves that group to disk. Since the object group 305 is still cached on the cache object store platform, if that group is subsequently needed, access to the group can be achieved quickly instead of having to go back to the source.

If write operations are occurring near the read operations, it is likely that there will be areas of the disk that are not being accessed. These areas may contain aged cache data objects that should be replaced. Accordingly, the cache object store writes to the "oldest" cache object groups and does a seek operation on back-to-back write accesses. New data is written to these "cold" areas. Since there are multiple possible targets (one per head) the object store determines the "best closest" area to write using the following ranking factors, in priority order: (i) write access immediately after another write access on the same head to substantially reduce latency; (ii) write access to disk free space rather than overwriting "old" cache object groups to maintain as much old data on the cache object store 300 as possible; (iii) write

access to large free or freeable spaces; (iv) write access to a space where no or minimal space is wasted at the end; and (v) write access to a disk location that has minimal added latency.

Also, the cache object store attempts to minimize head 322 movement by confining data caching in the secondary storage level 320 to a certain area, thereby creating "high object access traffic". Consequently, other areas of the disk are not accessed often and become cold. As an area of the secondary store becomes colder, its priority rises to a point where the management algorithm absorbs the delay required to move the heads to that cold area of the disk. Disk caching accesses then become concentrated in that area by minimizing head movement, thereby warming-up that area. This feature enables utilization of the entire geography of the disk.

According to the invention, the modified LRU algorithm may be advantageously used to implement a storage management policy that forces disk write operations to occur at these cold areas. Essentially, the algorithm cooperates with the directory 314 to provide a disk management policy that attempts to relocate the heads to the oldest cold area.

In order to maximize disk throughput, the secondary level cache 320 strives to keep multiple operations queued at a time. This may cause a slight inefficiency in read operations (since the next "best" read operation choice regarding parallel disk accesses is not always made). This also slows down the responsiveness to read accesses if two write operations are previously queued for execution. Ideally, a disk controller 380 signals the cache object manager 350 that the current disk operation is almost completed; that way a second operation may be subsequently queued and a better decision may be made with respect to that operation. Nevertheless, the cache object store may occasionally "waste" space due to set up/transfer times between previous operations; fundamentally, the intent is to keep the cache object store 300 constantly busy by maximizing the throughput and efficiency of the secondary level store 320.

While there has been shown and described an illustrative embodiment of a cache object store organized to provide fast and efficient storage of data as cache objects organized into cache object groups, it is to be understood that various other adaptations and modifications may be made within the spirit and scope of the invention. For example, it should be noted that each cache object/group is assigned a state as it transitions through various stages so that it can be managed by the cache object manager or, more particularly, the state machine implementation of the cache object manager. For instance, an object enters a state when it is first formed, it transitions to a new state when it exhausts its TTL, and another state is entered depending upon whether the object is in memory or moved to disk.

The foregoing description has been directed to specific embodiments of this invention. It will be apparent, however, that other variations and modifications may be made to the described embodiments, with the attainment of some or all of their advantages. Therefore, it is the object of the appended claims to cover all such variations and modifications as come within the true spirit and scope of the invention.

What is claimed is:

1. A cache object store organized to provide fast and efficient storage of data as cache objects organized into cache object groups, the cache object store comprising:
 - a primary-level store and a secondary-level store;
 - a cache grouper for logically associating cache objects into cache object groups; and

13

a cache object manager cooperatively coupled to the cache grouper and configured to implement an aging policy, the aging policy maintaining a first cache object groups on the primary-level store and a second cache object groups on the secondary-level store;

a cache directory manager cooperatively coupled to the cache object manager to implement a storage management policy that moves certain first cache object groups from the primary-level store for storage on the secondary-level store as second cache object groups, and the aging policy is a modified least recently used (LRU) algorithm that marks each cache object group with (i) a time of last access (TLA) factor within the cache object store, and (ii) a cost of reacquisition (COA) factor used to determine which cache object group to delete or move.

2. The cache object store of claim 1 further comprising a tertiary-level store cooperatively coupled to the primary and secondary level stores.

3. The cache object store of claim 2 wherein the tertiary-level store comprises bubble memory.

4. The cache object store of claim 1 further comprising an aging table structure maintained by the cache object manager to track the TLA and COA factors during implementation of the modified LRU by the cache object manager.

5. A cache object store, comprising:

- a primary-level store having random access memory;
- a secondary-level store coupled to the primary-level store, the secondary-level store having at least one non-volatile disk;
- a cache grouper for logically associating cache objects into cache object groups;
- a cache object manager cooperatively coupled to the cache grouper and configured to implement an aging policy that maintains a first cache object groups on the primary-level store, the aging policy further maintaining a second cache object groups on the secondary-level store; and,
- a cache directory manager cooperatively coupled to the cache object manager to implement a storage management policy that moves certain of the first cache object groups from the primary-level store for storage on the secondary-level store as the second cache object groups, the storage management policy substantially improving storage of the second cache object groups on the secondary-level store by avoiding fragmentation of the second cache object groups on the secondary-level store, and wherein the storage management policy moves the first cache object groups from the primary-level store to the secondary-level store by first searching the secondary-level store at a location in proximity to where the secondary-level store was last accessed.

6. The cache object store of claim 5 further comprising a list of cache object groups maintained by the cache object manager that specifies those first cache object groups, by cache object group size, to move from the primary-level store to the secondary-level store.

7. The cache object store of claim 6 wherein the cache directory manager comprises a memory directory submanager and a disk directory submanager adapted to manage organizations of the primary-level and secondary-level stores, respectively, to thereby implement the storage management policy.

8. A method for efficiently storing data as cache objects on a cache object store, the method comprising the steps of:

- providing a primary-level volatile store coupled to a secondary-level non-volatile store;

14

- organizing the cache objects into cache object groups; implementing an aging policy that maintains first, frequently-accessed cache object groups on the primary-level store and second, less-frequently-accessed cache object groups on the secondary-level store; and
- implementing a storage management policy that efficiently moves certain first cache object groups from the primary-level store for storage on the secondary-level store as second cache object groups, and the aging policy is a modified least recently used (LRU) algorithm for marking each cache object group with (i) a time of last access (TLA) factor indicating a frequency at which the cache object group is accessed within the cache object store, and (ii) a cost of reacquisition (COA) factor used to determine which cache object group to delete or move.

9. The method of claim 8 wherein the step of implementing the modified LRU aging policy further comprises the step of empirically determining the COA factor by measuring the turnaround time needed to acquire a cache object group, wherein the turnaround time is defined as time elapsed from when a request is issued to acquire the cache object group to when the cache object group is retrieved.

10. The method of claim 8 wherein the step of implementing the modified LRU aging policy further comprises the step of targeting those objects for movement or deletion based on relative computation of the TLA and COA factors.

11. A node for carrying out the method according to claim 8.

12. A computer network comprising at least one node for carrying out the method according to claim 8.

13. A computer-readable medium comprising: instructions for execution on a processor and data written thereon, said instructions and data containing information for the practice of the method of claim 8.

14. Electromagnetic signals traveling over a computer network comprising: said electromagnetic signals carrying information including instructions for execution on a processor for the practice of the method of claim 8.

15. A method for storing data as cache objects on a cache object store, the method comprising the steps of:

- providing a primary-level volatile store coupled to a secondary-level non-volatile store;
- organizing the cache objects into cache object groups; implementing an aging policy that maintains first, frequently-accessed cache object groups on the primary-level store and second, less-frequently-accessed cache object groups on the secondary-level store; and,
- implementing a storage management policy that moves certain first cache object groups from the primary-level store for storage on the secondary-level store as second cache object groups, the management policy by avoiding fragmentation to each certain cache object group on the secondary-level, non-volatile store, and wherein the management policy moves the first cache object groups from the primary-level store to the secondary-level store by first searching the secondary-level store at a location in proximity to where the secondary-level store was last accessed.

16. The method of claim 15 wherein the secondary-level, non-volatile store comprises a disk and wherein the step of implementing the storage management policy further comprises the steps of:

- conducting a search for a preconfigured level of free space around an area of the disk that is last accessed; and

15

upon discovering the preconfigured level of free space, storing the certain cache object group on the free space.

17. The method of claim 16 wherein the step of implementing the storage management policy further comprises the steps of:

if the preconfigured level of free space is not discovered, generating a list of all free spaces on the disk; and generating a list of available spaces on the disk.

18. The method of claim 17 wherein the step of implementing the storage management policy further comprises the steps of:

comparing combinations of certain cache object groups that fit the listed free spaces; and

if combinations of certain cache object groups are found that fit the free spaces, storing those certain cache object groups on the free disk spaces.

19. The method of claim 18 wherein the step of implementing the storage management policy further comprises the steps of:

if combinations of certain cache object groups are not found that fit the free spaces, comparing combinations of certain cache object groups that fit the listed available spaces; and

storing the remaining certain cache object groups on the available disk spaces.

20. The method of claim 19 wherein the step of comparing combinations of certain cache object groups that fit the listed available spaces comprises the step of calculating the available spaces by coalescing boundaries between the two lists of spaces.

21. A cache object store, comprising:

a primary-level store;

a secondary-level store coupled to the primary-level store;

a cache grouper for logically associating cache objects into cache object groups;

a cache object manager cooperatively coupled to the cache grouper and configured to implement an aging policy that maintains a first cache object groups on the primary-level store, the aging policy further maintain-

16

ing a second cache object groups on the secondary-level store; and,

a cache directory manager cooperatively coupled to the cache object manager to implement a storage management policy that moves certain of the first cache object groups from the primary-level store for storage on the secondary-level store as the second cache object groups, the storage management policy substantially improving storage of the second cache object groups on the secondary-level store by avoiding fragmentation of the second cache object groups on the secondary-level store, and wherein the storage management policy moves the first cache object groups from the primary-level store to the secondary-level store by first searching the secondary-level store at a location in proximity to where the secondary-level store was last accessed.

22. A method for storing data as cache objects on a cache object store, the method comprising the steps of:

providing a primary-level store coupled to a secondary-level;

organizing the cache objects into cache object groups;

implementing an aging policy that maintains first, frequently-accessed cache object groups on the primary-level store and second, less-frequently-accessed cache object groups on the secondary-level store; and,

implementing a storage management policy that moves certain first cache object groups from the primary-level store for storage on the secondary-level store as second cache object groups, the management policy by avoiding fragmentation to each certain cache object group on the secondary-level, non-volatile store, and wherein the storage management policy moves the first cache object groups from the primary-level store to the secondary-level store by first searching the secondary-level store at a location in proximity to where the secondary-level store was last accessed.

* * * * *

Webster's 1828 Dictionary, Electronic Version by Christian Technologies, Inc.

*Your word is
a lamp to my
feet And a
light to my
path.*

Psa 119:105



Available on
CDROM



Hardcopy



American
Student's
Package



The Holy
Scriptures

Enter word to search for:

Search

List words in dictionary

PROV'DER, n. One who provides, furnishes or supplies; one that procures what is wanted.

[Previous page in the dictionary] [Next page in the dictionary]

provider is also found in 3 definitions:

- cater
- caterer
- cateress

[Return to CTT home page]

*Christian Technologies, Inc.
PO BOX 2201, Independence, MO 64055
Phone: 1-800-366-8320 E-Mail: info@christiantech.com*

(c) 2002 Christian Technologies, Inc.

Webster's 1828 Dictionary, Electronic Version by Christian
Technologies, Inc.

*Your word is
a lamp to my
feet And a
light to my
path.*

Psa 119:105



Available on
CDROM



Hardcopy



American
Student's
Package



The Holy
Scriptures

Enter word to search for:

Search

List words in dictionary

AGGREGATOR, n. He that collects into a whole or mass.

[\[Previous page in the dictionary\]](#) [\[Next page in the dictionary\]](#)

[\[Return to CTI home page\]](#)

*Christian Technologies, Inc.
PO BOX 2201, Independence, MO 64055
Phone: 1-800-366-8320 E-Mail: info@christiantech.com*

(c) 2002 Christian Technologies, Inc.

Webster's 1828 Dictionary, Electronic Version by Christian Technologies, Inc.

*Your word is
a lamp to my
feet And a
light to my
path.*

Psa 119:105



Available on
CDROM



Hardcopy



American
Student's
Package



The Holy
Scriptures

Enter word to search for:

Search

List words in dictionary

PROV'DER, n. One who provides, furnishes or supplies; one that procures what is wanted.

[Previous page in the dictionary] [Next page in the dictionary]

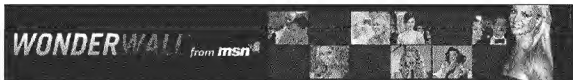
provider is also found in 3 definitions:

- cater
- caterer
- cateress

[Return to CTT home page]

Christian Technologies, Inc.
PO BOX 2201, Independence, MO 64055
Phone: 1-800-366-8320 E-Mail: info@christiantech.com

(c) 2002 Christian Technologies, Inc.




WONDERWALL from msn

GET YOUR
CELEBRITY
PICS AND
STORIES >>>

[encarta](#)
[greeting cards](#)
[more](#)

[MSN home](#)
[Mail](#)
[My MSN](#)
[Sign in](#)



[Search Encarta](#)
[Web](#)

[Home](#)
[Encyclopedia](#)
[Dictionary](#)
[Atlas](#)
[K-12 Success](#)
[College & Grad School](#)
[Degrees & Training](#)
[Quizzes](#)
[More](#)

Dictionary

Find

provider

in

Dictionary

Dictionary **Thesaurus** **Translations**

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- ↑

proverbial

Proverbs

provide

provided

Providence

providence

provident

providential

► **provider**

providing

provincia

Provincetown

provincial

Provincial Council

provincial court

provincial police

provincialism
- ↓

provider

► provider

pro-vi-der [prə vīdər] (*plural* pro-vi-ders)

noun

Definition:

1. **supplier of service:** an organization or company that provides access to a service or system such as a cellular phone, cable, or computer network

- an Internet provider
- a provider

2. **supplier of support:** somebody who provides material support for somebody or something, especially a family

Advertisement



Get a one-month FREE* trial pair.

Also on Encarta

- 10 tips for building your English vocabulary
- Alternatives to the M.B.A.
- GPS for students
- Saving for college
- Presidential Myths Quiz
- Coffee break: Recharge your brain

Make your dreams come true with EnglishTown.

Our Partners

- ApplyWise Online College Admissions Counseling
- The Princeton Review
- Tutor.com: Live Help When You Get Stuck

Also on MSN

- "Treasure Quest": recovering sunken loot on the Discovery Channel
- MSN Shopping: Best books for the season
- MSN Careers: Nail that job interview
- Today's news on MSNBC

Print Preview

See pronunciation key

Search for "provider" in all of MSN Encarta

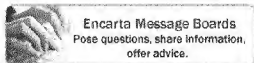
E-mail this entry

Blog about this entry on MSN Spaces

Download the MSN

Encarta Right-Click
Dictionary

Encarta® World English Dictionary [North American Edition] © & (P)2009
Microsoft Corporation. All rights reserved. Developed for Microsoft by
Bloomsbury Publishing Plc.



MSN Shopping



Electronics

Laptops

HDTVs

Smartphones


Digital cameras


[MSN Privacy](#) [Legal](#) [Advertise](#) [Newsletter](#) [Worldwide](#)

[Feedback](#) [Help](#)

© 2008 Microsoft


Microsoft





[encarta](#)
[greeting cards](#)
[more](#)

[MSN home](#)
[Mail](#)
[My MSN](#)
[Sign in](#)



[Home](#)
[Encyclopedia](#)
[Dictionary](#)
[Atlas](#)
[K-12 Success](#)
[College & Grad School](#)
[Degrees & Training](#)
[Quizzes](#)
[More](#)

Dictionary

Find

aggregator

in

Dictionary

[Dictionary](#)
[Thesaurus](#)
[Translations](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z



aggravated
 aggravation
 aggregate
 aggregate demand
 aggregate expenditures
 aggregate income
 aggregate output
 aggregate supply
aggregator
 aggress
 aggression
 aggressive
 aggressive growth fund
 aggressor
 aggrieve
 agro
 aggrupation



aggregator

ag-gre-ga-tor [ággrə gàytər] (plural
 ag-gre-ga-tors)

noun

Definition:

1. **somebody or something bringing others together:** a person, organization, or thing that brings different things or people together, into a total, mass, or whole

2. **BUSINESS organization that gathers customers:** an organization that unites customers into a buying group to obtain better prices

Advertisement

True



Also on Encarta

- 10 tips for building your English vocabulary
- Alternatives to the M.B.A.
- GPS for students
- Saving for college
- Presidential Myths Quiz
- Coffee break: Recharge your brain

Make your dreams
come true with
Englishtown.

Our Partners

- ApplyWise Online College Admissions Counseling
- The Princeton Review
- Tutor.com: Live Help When You Get Stuck

Also on MSN

- "Treasure Quest": recovering sunken loot on the Discovery Channel
- MSN Shopping: Best books for the season
- MSN Careers: Nail that job interview
- Today's news on MSNBC

Print Preview

See pronunciation
keySearch for
"aggregat..." in all of
MSN Encarta

[E-mail this entry](#)

[Blog about this entry
on MSN Spaces](#)

[Download the MSN
Encarta Right-Click
Dictionary](#)

Encarta® World English Dictionary [North American Edition] © & (P)2009
Microsoft Corporation. All rights reserved. Developed for Microsoft by
Bloomsbury Publishing Plc.



Encarta Message Boards
Pose questions, share information,
offer advice.

MSN Shopping



Spring Fashion for Him

[Jeans](#)
[Swimwear](#)
[Casual Shoes](#)
[Sunglasses](#)

[MSN Privacy](#) [Legal](#) [Advertise](#) [Newsletter](#) [Worldwide](#)

[Feedback](#) [Help](#)

© 2008 Microsoft

Microsoft

Gateway

From Wikipedia, the free encyclopedia

A **gateway** is a point of entry or exit at which a **gate** may be hung.

Gateway may also refer to:

Computer terminology

- Gateway (telecommunications), a computer or a network that allows or controls access to another computer or network
- Gateway (web page), a webpage designed to attract visitors and search engines to a particular website
- Payment gateway, the software interface between a web-based shopping cart and a merchant account
- Gateway (computer program), a link between two computer programs allowing them to share information and bypass certain protocols on a host computer
- Residential gateway, a home networking device

Companies

- Gateway, Inc. (formerly Gateway 2000), a computer manufacturer
- Gateway, a former name of the UK supermarket chain now rebranded as Sumerfield
- Gateway Building Society, a UK building society bought in 1988 by the Woolwich Building Society
- Gateways Books and Tapes, a book and audio tape retailer bought in 1988 by Books-A-Million
- Gateway Newstands, a Canadian convenience store chain located in large office buildings, shopping centres, public places, and transit stations in the United States and Canada

Churches

- Gateway Church (Texas), a mega-church located in the Dallas-Fort Worth Metroplex
- Gateway Church, a church located in Victoria, Australia

Geography

Africa

South Africa

- Gateway Theatre of Shopping, a shopping centre in Durban

Australia

- Gateway Bridge, Brisbane, the most eastern crossing of the Brisbane River, Brisbane
- Gateway Motorway, a freeway in Brisbane

Asia

Hong Kong

- The Gateway, Hong Kong, a shopping centre in Tsim Sha Tsui, Kowloon, Hong Kong

India

- Gateway of India, Bombay

Singapore

- Gateway Monorail Station, a station on the Sentosa Monorail
- The Gateway, Singapore, a twin-building office complex

Contents

- 1 Computer terminology
- 2 Companies
- 3 Churches
- 4 Geography
 - 4.1 Africa
 - 4.1.1 South Africa
 - 4.2 Australia
 - 4.3 Asia
 - 4.3.1 Hong Kong
 - 4.3.2 India
 - 4.3.3 Singapore
 - 4.3.4 Turkestan
 - 4.4 Europe
 - 4.4.1 Poland
 - 4.4.2 Romania /Serbia
 - 4.5 North America
 - 4.5.1 Canada
 - 4.5.2 United States : Communities
 - 4.5.3 United States : Landmarks and structures
- 5 Music
- 6 Film
- 7 In fiction
- 8 Other
- 9 See also

Turkestan

- Iron Gate (Central Asia)

Europe

Poland

- Iron Gate, Warsaw

Romania /Serbia

- Iron Gate

North America

Canada

- Gateway Boulevard, a major street in Edmonton
- Gateway Station on the Vancouver SkyTrain's Expo Line
- Greater Vancouver Gateway Program, an infrastructure project for Greater Vancouver

United States : Communities

- Gateway, Alaska, census-designated place
- Gateway, Arkansas, town
- Gateway, California
 - Gateway, Los Angeles County, California
 - Gateway, Nevada County, California
 - Gateway, San Diego, California, neighborhood
- Gateway, Colorado, unincorporated town
- Gateway, Florida, census-designated place

United States : Landmarks and structures

- Gateway International Raceway, a racetrack in Madison, Illinois
- Gateway Arch, part of the Jefferson National Expansion Memorial in St. Louis
- Gateway Region, the most urbanized part of northeastern New Jersey, including Newark and Jersey City
- Gateway National Recreation Area, a string of parks and beaches in New York City and New Jersey
- Westfield Gateway, an enclosed shopping mall in Lincoln, Nebraska
- Gateway Station (Charlotte, North Carolina), a proposed intermodal transit center in Charlotte, North Carolina
- The Gateway Sports and Entertainment Complex (more commonly known as "Gateway"), a stadium complex in Cleveland, Ohio, containing Progressive Field and Quicken Loans Arena
- The Gateway District, a large open air retail, residential and office complex in Salt Lake City

Music

- Gateway (band) a jazz trio featuring John Abercrombie, Jack DeJohnette and Dave Holland
 - *Gateway* (Gateway album) a 1976 album released by the band on ECM Records
- *Gateway* (album), an album by stoner metal band Bongzilla

Film

- *Gateway* (film), a 1938 dramatic film directed by Alfred L. Werker

In fiction

- Common synonym for portal
- *Gateway* (novel), a 1977 novel by Frederik Pohl, the first book of the Heechee series and winner of the Hugo Award and Nebula Award
 - *Gateway* (computer game), two adventure games based on the novel have been created by Legend Entertainment
- Gateway (comics), a supporting character in Marvel's *X-Men* series

Other

- Gateway drug theory, a hypothesis on drug use
- Gateway Rehabilitation center, a rehabilitation center for drug and alcohol addicts, in Aliquippa, Pennsylvania, USA
- "Gateway" in oceanography, ocean currents in the global thermohaline circulation

- "Gateway" in aviation, the start or the termination for airlines at airports
- Gateway Technology, a cloning system in molecular biology

See also

- Gateways
- Gateway Bridge (disambiguation)
- Gateway Center
- Gateway Mall
- Gateway Theatre

Retrieved from "http://en.wikipedia.org/wiki/Gateway"

Categories: Disambiguation pages | Place name disambiguation pages

- This page was last modified on 3 June 2010 at 16:28.
- Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. See Terms of Use for details. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.
- Privacy policy
- About Wikipedia
- Disclaimers

Gateway (telecommunications)

From Wikipedia, the free encyclopedia

In telecommunications, the term **gateway** has the following meaning:



Juniper SRX210 service gateway

- In a communications network, a network node equipped for interfacing with another network that uses different protocols.
 - A gateway may contain devices such as protocol translators, impedance matching devices, rate converters, fault isolators, or signal translators as necessary to provide system interoperability. It also requires the establishment of mutually acceptable administrative procedures between both networks.
 - A protocol translation/mapping gateway interconnects networks with different network protocol technologies by performing the required protocol conversions.
- Loosely, a computer configured to perform the tasks of a gateway. For a specific case, see default gateway.

Routers exemplify special cases of gateways.

Gateways, also called **protocol converters**, can operate at any layer of the OSI model. The job of a gateway is much more complex than that of a router or switch. Typically, a gateway must convert one protocol stack into another.

Contents

- 1 Details
- 2 Internet-to-Orbit Gateway
- 3 Examples
- 4 References
- 5 Sources
- 6 See also

Details

A gateway is a network point that acts as an entrance to another network. On the Internet, a node or stopping point can be either a gateway node or a host (end-point) node. Both the computers of Internet users and the computers that serve pages to users are host nodes, while the nodes that connect the networks in between are gateways. For example, the computers that control traffic between company networks or the computers used by internet service providers (ISPs) to connect users to the internet are gateway nodes.

In the network for an enterprise, a computer server acting as a gateway node is often also acting as a proxy server and a firewall server. A gateway is often associated with both a router, which knows where to direct a given packet of data that arrives at the gateway, and a switch, which furnishes the actual path in and out of the gateway for a given packet.

On an IP network, clients should automatically send IP packets with a destination outside a given subnet mask to a network gateway. A subnet mask defines the IP range of a network. For example, if a network has a base IP address of 192.168.0.0 and has a subnet mask of 255.255.255.0, then any data going to an IP address outside of 192.168.0.X will be sent to that network's gateway. While forwarding an IP packet to another network, the gateway might or might not perform Network Address Translation.

A gateway is an essential feature of most routers, although other devices (such as any PC or server) can function as a gateway.

Most computer operating systems use the terms described above. A computer running Microsoft Windows however describes this standard networking feature as Internet Connection Sharing, which will act as a gateway, offering a connection between the Internet and an internal network. Such a system might also act as a DHCP server. Dynamic Host Configuration Protocol (DHCP) is a protocol used by networked devices (clients) to obtain various parameters necessary for the clients to operate in an Internet Protocol (IP) network. By using this protocol, system administration workload greatly decreases, and devices can be added to the network with minimal or no manual configurations.

Internet-to-Orbit Gateway

An Internet to orbit gateway (I2O) is a machine that acts as a connector between computers or devices connected to the Internet and computer systems orbiting the earth, like satellites or even manned spacecrafts. Such connection is made when the I2O establishes a stable link between the spacecraft and a computer or a network of computers on the Internet, such link can be control signals, audio frequency, or even visible spectrum signals.

Project HERMES is the first of this kind of devices to become operative. The HERMES-A/MINOTAUR Space Flight Control Center^[1] became operative on June 6 2009 and was operated by representatives of 34 countries on the UNOOSA Symposium of Small Satellites for Sustainable Development^[2] in Graz, Austria on September 10, 2009. Project HERMES is an initiative of the Ecuadorian Civilian Space Agency and has a maximum coverage of 22,000 km. HERMES-A is supposed to be the first gateway of a network of five covering all south America. HERMES-A/MINOTAUR is not only capable of data transmission but voice also.

Project GENSO is an initiative from NASA and ESA, and it is expected to begin operations on April 2010, it is supposed to have worldwide coverage.

Examples

- A very popular example is connecting a Local Area Network or Wireless LAN to the Internet or other Wide Area Network. In this case the gateway connects a LAN to the provider-specific network which in turn connects to the Internet. In the case of a home, this gateway is called a residential gateway.
- MainWay is the Bull brand for a gateway which connects DSA to TCP/IP telecom

References

- ↑ http://cxa.co/bp25
- ↑ http://www.unoosa.org/pdf/sap/2009/graz/Programme_3September.pdf

Sources

- Federal Standard 1037C
- MIL-STD-188

See also

- Router
- Subnet

Retrieved from "http://en.wikipedia.org/wiki/Gateway_(telecommunications)"

Categories: Networking hardware | Routers | Internet architecture | Videotelephony

-
- This page was last modified on 7 July 2010 at 21:09.
 - Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. See Terms of Use for details. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.
 - Privacy policy
 - About Wikipedia
 - Disclaimers

Merriam-Webster
OnLine

Home
Premium Services
Downloads
Word of the Day
Word Games
Open Dictionary
Spelling Bee Hive
Online Store
Help
About Us

Also Visit: [Unabridged](#) | [Visual](#) | [Britannica Online Encyclopedia](#) | [ESL](#) | [Learner's](#) | [for Kids](#) | [Word Central](#)

[Dictionary](#) | [Thesaurus](#) | [Spanish/English](#) | [Medical](#)

[Search](#)

gateway

Join Us On [Twitter](#)

2 entries found.

gateway (noun)
gateway drug (noun)

Ads by Google

[Fieldbus OEM modules](#)

Build-in modules and gateways for field devices for process control
[www.fintao](#)

Main Entry: **gate·way** 4

Pronunciation: ˈ-,wā

Function: *noun*

Date: 1707

1 : an opening for a gate

2 : [GATE](#) 4a

Learn more about "gateway" and related topics at
[Britannica.com](#)

Ads by Google

[Fieldbus OEM modules](#)

Build-in modules and gateways for field devices for process control
[www.fint.no](#)

[Router Switches Avail.](#)

All hardware & Router Products Contact Us Today
[www.TSsystemsinc.com](#)

[Pronunciation Symbols](#)

Share this entry:



Link to this page:

gateway

Cite this page:

MLA Style

"gateway." Merriam-Webster Online Dictionary 2010.
Merriam-Webster Online. 31 July 2010.
<<http://www.merriam-webster.com/dictionary/gateway>>

APA Style

gateway. (2010). In Merriam-Webster Online Dictionary.
Retrieved July 31, 2010, from <http://www.merriam-webster.com/dictionary/gateway>

Do These Words
Stump You
Too?



Find Out in Merriam-Webster's
TOP 10 LISTS

gateway

[Go](#)

Search "gateway" in:

- * [Thesaurus](#)
- * [Spanish/English](#)
- * [Medical Dictionary](#)
- * [Open Dictionary](#)

Browse words next to:

- * [gateway](#)

Browse the Dictionary:

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#) [Y](#) [Z](#)

[Products](#) [Premium Services](#) [Company Info](#) [Contact Us](#) [Advertising Info](#) [Privacy Policy](#)

© 2010 Merriam-Webster, Incorporated



6950 Enterprise Gateway Server
USER'S GUIDE



PN: 961-047-091
Revision D
September 1999

► **NOTICE**

The information contained herein is proprietary and is provided solely for the purpose of allowing customers to operate and service Intermec manufactured equipment and is not to be released, reproduced, or used for any other purpose without written permission of Intermec.

Disclaimer of Warranties. The sample source code included in this document is presented for reference only. The code does not necessarily represent complete, tested programs. The code is provided **"AS IS WITH ALL FAULTS."** **ALL WARRANTIES ARE EXPRESSLY DISCLAIMED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.**

We welcome your comments concerning this publication. Although every effort has been made to keep it free of errors, some may occur. When reporting a specific problem, please describe it briefly and include the book title and part number, as well as the paragraph or figure number and the page number.

Send your comments to:
Intermec Technologies Corporation
Publications Department
550 Second Street SE
Cedar Rapids, IA 52401

INTERMEC and NORAND are registered trademarks and ENTERPRISE WIRELESS LAN, UAP, and UNIVERSAL ACCESS POINT are trademarks of Intermec Technologies Corporation.

© 1996 Intermec Technologies Corporation. All rights reserved.

Acknowledgments

AS/400 and *IBM* are registered trademarks of International Business Machines Corporation.

DEC, *VAX*, and *VT220* are registered trademarks of Digital Equipment Corporation.

UNIX is a registered trademark of UNIX System Laboratories, Inc.

▼ **CAUTION:**

Intermec Technologies Corporation suggests you buy cables from us to connect with other devices. Our cables are safe, meet FCC rules, and suit our products. Other cables may not be tested. They may cause problems from electrostatic discharge or induced energy. Our warranties do not cover loss, injury, or damage from other cables.

[illegible]

Read This First	1-1
Scope	1-1
Purpose	1-1
Assumptions	1-1
Limitation	1-2
Content Summary	1-2
Related Publications	1-3

Introduction	2-1
About the 6950 EGS	2-1
Quick Start	2-2
Open the Box and Inspect Contents	2-4
Customer Support	2-4
Factory Service	2-5
Customer Response Center	2-5
Web Site	2-5
Bulletin Board Service	2-6
Specifications	2-6

Installation	3-1
Site Consideration	3-2
Connecting an Access Point or Radio Base	3-3
Connecting a Controller or Gateway	3-4
RC3250 Network Controller	3-5
RCB4030 Base/Controller	3-6
RC4030E Gateway	3-7
Front Panel	3-8
Back Panel	3-9
Location	3-12
Mounting Brackets	3-12

Cabling	3-14
Collecting the Equipment	3-14
10BASE2 Components	3-14
10BASE-T or 100BASE-T Component	3-15
Connecting to Ethernet	3-16
10BASE2	3-16
End of Segment	3-16
Middle of Segment	3-16
10BASE-T or 100BASE-T	3-18
Applying Power	3-20
 SECTION 4	
Connecting to the 6950 EGS	4-1
Telnet	4-2
Video Monitor and Keyboard	4-4
Dumb Terminal	4-5
 SECTION 5	
Configuring Network Options	5-1
Required Information	5-1
Getting Started	5-3
Using the IP Menu	5-4
Configuring IP Addresses	5-6
Configuring Host Names	5-19
Adding a Host	5-20
Deleting a Host	5-21
Viewing IP Addresses and Host Names	5-21
Pinging a Host	5-23
Opening a Telnet Session	5-24
Setting Up the RF Units	5-25
Setting the Date and Time	5-28
 SECTION 6	
Configuring Radio Frequency Communication	6-1
Configuring Through an Ethernet Connection	6-2
Configuring Through an RS-232 Connection	6-6

SECTION 7

Halting, Rebooting, or Restarting the 6950 EGS	7-1
Halting the 6950 EGS	7-3
Rebooting the 6950 EGS	7-4
Restarting the 6950 EGS	7-6
Logging in to the Host	7-7
VT220 Terminal Emulation	7-7
TN3270 or TN5250 Terminal Emulation	7-9

SECTION 8

Monitoring the System	8-1
System Monitor	8-2
Logs	8-4
Viewing the Log File	8-5
Watching the Log File	8-7
Clearing the Log File	8-7
RS-232 Log Debug Levels	8-8
Debug Level=0	8-8
Debug Level=1	8-9
Debug Level=2	8-10
Ethernet Log	8-12
Debug Level=0	8-12
Debug Level=1	8-12
Debug Level=2	8-13
Abnormal Conditions	8-14
Time Test	8-15

SECTION 9

Updating the 6950 EGS Software	9-1
---	------------

SECTION 10

Host Application Program	10-1
General	10-1
Native Sockets Overview	10-2
Supported Products	10-2
Communication Basics	10-2
Host Programming	10-3
Coding the Application	10-4
Response Formats	10-5
Error Messages	10-5

Normal Communication	10-6
Coding the Application	10-6
NORAND® Native Syntax for Wireless Stations	10-6
Write Display	10-6
Other Commands and Responses	10-7
Native Mode Responses from the 6950 EGS	10-8
Configuration	10-9
Demonstration Program	10-12

INDEX Index-1

FIGURES

Figure 3-1 6950 EGS and Access Point or Radio Base Directly Connected	3-3
Figure 3-2 6950 EGS and Access Point Linked to Host Via Router	3-4
Figure 3-3 RC3250 Network Controller Connected to 6950 EGS	3-5
Figure 3-4 RC3250 Network Controller Attached with Modem to 6950 EGS	3-6
Figure 3-5 6950 EGS Connected to RCB4030 Base/Controller	3-7
Figure 3-6 6950 EGS Connected to RC4030E Gateway ..	3-7
Figure 3-7 Front Panel	3-8
Figure 3-8 Back Panel of Current Model of 6950 EGS ..	3-10
Figure 3-9 Back Panel of Early Model of 6950 EGS	3-11
Figure 3-10 T-connector	3-14
Figure 3-11 Cable Terminator	3-15
Figure 3-12 Cable With RJ45 Plugs	3-15
Figure 3-13 6950 EGS at End of 10BASE2 Segment	3-17
Figure 3-14 6950 EGS in Middle of 10BASE2 Segment .	3-18
Figure 3-15 6950 EGS Connected to 10BASE-T or 100BASE-T	3-19
Figure 3-16 AC Power Connection	3-21
Figure 5-1 Default Gateway to Host	5-10
Figure 5-2 Static Route to Host	5-12

TABLES

Table 10-1 Abbreviated Commands	10-4
Table 10-2 Write Display Error Codes	10-5
Table 10-3 Abbreviated Commands	10-7
Table 10-4 Codes in Wireless Station Number Field	10-8

Section 1

Read This First

Scope

This document covers installation, configuration, and upgrade of the 6950 Enterprise Gateway Server (EGS). It also describes the information available through logs to assist in isolating communication problems.

► **NOTE:**

We continue to use the NORAND[®] name in this guide. It was retained to eliminate confusion since the 6950 EGS operates with wireless network devices that still carry the NORAND label.

Purpose

To aid the person installing, configuring, upgrading, or troubleshooting the 6950 EGS.

Assumptions

This guide assumes you already know how to cut, splice, and attach connectors to cabling. It also assumes you know the basics of internet addressing and TCP/IP.

Limitation

The 6950 EGS is a conduit between wireless stations and a host application. We provide minimal information about writing host applications. Developers can find extensive assistance about writing to and reading from the wireless stations by consulting the programmer's guide for their data stream. We also offer a guide for addressing wireless stations in their native mode.

Content Summary

This guide tells:

- ▶ How to perform a quick start, what comes with the 6950 EGS, optional mounting brackets, and where to call for support — *Section 2, "Introduction"*
- ▶ How to install the 6950 EGS and other devices on your Ethernet network — *Section 3, "Installation"*
- ▶ How to connect to the 6950 EGS via telnet, a video monitor and keyboard, or a dumb terminal — *Section 4, "Connecting to the 6950 EGS"*
- ▶ How to configure your network-specific parameters, including IP addresses, host names and addresses, and static routing — *Section 5, "Configuring Network Options"*
- ▶ How to configure RF communications — *Section 6, "Configuring Radio Frequency Communication"*
- ▶ How to stop, reboot, or restart the 6950 EGS — *Section 7, "Halting, Rebooting, or Restarting the 6950 EGS"*
- ▶ How to monitor the 6950 EGS and the system — *Section 8, "Monitoring the System"*

- ▶ How to update the 6950 EGS's software and what information to save before you begin — *Section 9, "Updating the 6950 EGS Software"*
- ▶ The basic information you must know to develop a host application or adapt an existing one — *Section 10, "Host Application Program"*

Related Publications

To order a printed manual, contact your Intermec Sales Representative. Several online manuals are also available in Portable Document Format (PDF) on the Intermec web site. The list of online manuals is at:

<http://www.intermec.com/manuals/english.htm>

Or, choose "Products" then "Manuals" and "English" from the opening page.

You must download the free Adobe Acrobat Reader to view the PDF manuals. Instructions are at:

<http://www.intermec.com/manuals/manuals.htm#reader>

Following are related INTERMEC manuals and part numbers (PN).

- ▶ *6710 Access Point User's Guide* PN: 961-047-081
- ▶ *2100 Universal Access Point User's Manual*
PN: 0671150
- ▶ *RB4030 Radio Base User's Guide* PN: 961-047-057
- ▶ *RCB4030 Spread Spectrum Base and Base / Controller User's Guide* PN: 961-047-075
- ▶ *RC3240 / RC3250 Network Controller User's Guide*
PN: 961-047-040

- ▶ User's guide for any base radio the RC3250 Network Controller supports.
- ▶ User's guides for these wireless stations:
 - 5055 PN: 961-054-017
 - 6400 PN: 961-047-093
 - 6500/6550 PN: 961-047-099
 - RT1100 PN: 961-047-069
 - RT1700 PN: 961-047-068
 - RT5900 PN: 961-047-121
- ▶ Programmer guides for writing the host application for the wireless stations:
 - 3270 Terminal Emulation Programmer's Reference Guide* PN: 977-047-040
 - 5250 Terminal Emulation Programmer's Reference Guide* PN: 977-047-039
 - Native Terminal Emulation Asynchronous Programmer's Reference Guide* PN: 977-047-038
 - UNIX Network Programming* by W. Richard Stevens ISBN 0-13-94876-1
 - VT220 Terminal Emulation Programmer's Reference Guide* PN: 977-047-037

Section 2

Introduction

About the 6950 EGS

The 6950 EGS is a communication gateway between applications running on your host computer or computers and an Enterprise Wireless LAN by INTERMEC®. It connects your hand-held wireless stations and enterprise host via a TCP/IP telnet session. The 6950 EGS supports communications from wireless stations to multiple host applications.

The Enterprise Wireless LAN is the portion of the network that handles communication between INTERMEC® wired and wireless products. A wireless network includes the 6950 EGS; RC3250 Network Controller, RCB4030 Base/Controller, or RC4030E Gateway; 2100 UAPs, 6710 Access Points, or RB4030 Base Radios; and wireless stations.

The 6950 EGS connects to the RF network by one of these methods:

- ▶ A 2100 UAP or 6710 Access Point on the Enterprise Wireless LAN, connected (without a controller) via one of its Ethernet ports to the Ethernet network at a remote location from the 6950 EGS.
- ▶ An RB4030 Radio Base connected (without a controller) via one of its Ethernet ports to the Ethernet network at a remote location from the 6950 EGS.

- ▶ An RCB4030 Base/Controller connected through the 6950 EGS serial (COM1) port to its host port.
- ▶ A RC3250 Network Controller connected through the 6950 EGS serial RS-232 port (COM1) to its host port. The base radio connects to the network controller. This method supports UHF radio systems using RS-485 or RS-232 communication with the controller.

The 6950 EGS simultaneously supports 3270, 5250, and VT220 terminal emulation by the wireless stations. The wireless station's firmware sets the type of terminal emulation. Messages between the wireless station and the 6950 EGS set the communication method.

Quick Start

- ▶ **NOTE:** *For help with configuring the wireless station, RB4030 Base Radio, or RC3250, RC4030, or RCB4030 serial controller, refer to the device's user guide for instructions.*

Enterprise Wireless LAN Systems (Non-RB4030 Base Radio) Over Ethernet

(Page 3-3 shows an example of this type of system.)

1. Set the wireless stations' host name to "HOST".
2. Set the wireless stations' LAN ID to match the LAN ID for the 2100 Universal Access Point (UAP) or 6710 Access Point.
3. Set the wireless stations' terminal emulation type to 3270, 5250, or VT220.
4. If running 3270 or 5250, enable the wireless stations' "Telnet" option.
5. Configure the 6950 EGS's IP setup (*see Section 5 for help*).

SST Systems (RB4030 Base Radio) Over Ethernet

(Page 3-3 shows an example of this type of system.)

1. Set the wireless stations' host name to "HOST".
2. Set the wireless stations' LAN ID to 0 (zero).
3. Set the wireless stations' terminal emulation type to 3270, 5250, or VT220.
4. If running 3270 or 5250, enable the wireless stations' "Telnet" option.
5. Set one RB4030 Base Radio to be the "root" by setting dip switches 5 or 6 (or both).
6. Configure the 6950 EGS's IP setup (*see Section 5 for help*).

UHF or Asynchronous (RS-232) Systems

(Pages 3-5 through 3-7 show examples of these systems.)

1. Configure the RC3250, RC4030, or RCB4030 serial controller.
2. Configure the 6950 EGS to be "RS-232," and set the RS-232 settings to match those of the serial controller (*see Section 6 for help*).
3. Set the wireless stations' host name to match the serial controller's host name.
4. Set the wireless stations' LAN ID to match the serial controller's LAN ID.
5. Set the wireless stations' terminal emulation type to VT220.
6. Disable the wireless stations' "Telnet" option.
7. Configure the 6950 EGS's IP setup (*see Section 5 for help*).

Open the Box and Inspect Contents

The box should have one 6950 EGS, a 7 foot ac power cable, and (if you ordered an internal modem) a telephone cord with RJ11 plugs. If you ordered an optional mounting bracket, it is packaged separately.

1. Remove the 6950 EGS from its shipping container and inspect it for damage.
2. If it has been damaged in shipping, record the model number, part number, and serial number of the damaged unit.
3. Save any paperwork pertaining to the shipment, and immediately notify the transport company of the damaged item(s). Follow their instructions for filing a claim on the damaged item.

Customer Support

Customer Support's on-going objective is to provide quality support to all of our customers worldwide.

Factory Service

If your unit is faulty, you can ship it to the nearest authorized Service Center for factory-quality service. The addresses and telephone numbers are included in the Warranty Card shipped with your product.

Customer Response Center

The Customer Response Center (technical support) telephone number is 800-755-5505 (U.S.A. or Canada) or 425-356-1799. The facsimile number is 425-356-1688. Email is support@intermec.com.

If you email or fax a problem or question include the following information in your message: your name, your company name and address, phone number and email to respond to, and problem description or question (the more specific, the better). If the equipment was purchased through a Premier Solution Partner, please include that information.

Web Site

The Customer Support File Libraries, including Hot Tips and Product Awareness Bulletins, are available on the Internet. New users start at the Intemec web site: www.intermec.com. Choose "Support," then "Product Support," then "Conference Area." Look on the main page for a link to register new customers.

Bulletin Board Service

The Customer Support Bulletin Board (BBS), maintained by the Norand Mobile Systems Division of Intermec Technologies Corporation, provides software and documentation:

- ▶ **Phone number:** 319-369-3515 (14.4 Kbps modem)
319-369-3516 (28.8 Kbps modem)
- ▶ **Protocol:** Full duplex, ANSI or ANSI-BBS; 300 to 28,800 bps; v.32bis; 8 bits, no parity, 1 stop bit. *For high-speed modems, disable XON/XOFF and enable RTS/CTS.*

This is the same location available via the web site. If your web access uses high-speed phone lines, the web interface provides a faster response.

Specifications

For information about electrical and environmental specifications for the 6950 EGS, refer to the specifications provided with it.

Section 3

Installation



You must successfully establish two connections with the 6950 EGS after it powers up — one to the wireless equipment and one to the host. The 6950 EGS supports communication from wireless stations to multiple host applications. To establish connections, do the following:

- ▶ Connect the Ethernet port on your 6950 EGS to the network.
- ▶ Connect other network devices as follows:
 - ▶ Connect a 2100 UAP, 6710 Access Point, or RB4030 Base Radio through a direct connection to the Ethernet network. These devices do not require a controller.
 - ▶ Connect an RCB4030 Base/Controller, RC4030E Gateway, or RC3250 Network Controller through an RS-232 serial connection to the 6950 EGS. An RB4030 Base Radio or RB3000 Base Station directly connects to the controller.

This section describes some different configurations for the 6950 EGS and then shows how to install each one.

Site Consideration

Consider the following when preparing for the 6950 EGS.

- ▶ **Security:** Isolate the 6950 EGS from anyone who might tamper with it.
- ▶ **Environment:** Use an electrical circuit that does not already carry a heavy load. Protect the power switch so the 6950 EGS cannot be accidentally bumped and turned off. Make sure the room is well-ventilated to protect from overheating.
- ▶ **Access:** Place it where you can easily check or change the cables and see the “Power” and “HD” lights.
- ▶ **Cables:** Ensure they are defect free. Keep cable lengths short to reduce electrical interference.
- ▶ **Fault tolerance:** Consider an uninterruptable power supply. If the 6950 EGS serves a critical function, consider a spare.

Intermec strongly recommends that Intermec or certified providers conduct a site survey to determine the ideal locations for all of your network components. A proper site survey requires special equipment and training. A site survey provides an installation recommendation that addresses various factors that can affect the performance of your Enterprise Wireless LAN.

Connecting an Access Point or Radio Base

Figure 3-1 shows a configuration where the 6950 EGS and the radio link have been independently attached to the Ethernet network.

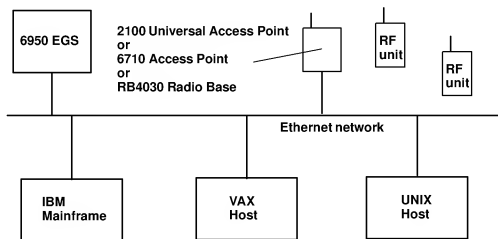


Figure 3-1
6950 EGS and Access Point or Radio Base Directly Connected

In this example, access points attached to the Ethernet backbone communicate with wireless stations with 2.4 GHz OpenAir or 900 MHz radios.

Figure 3-2 shows the 6950 EGS and access point connected to an IBM host through a router. The AS/400 host could be the default host supplying an application the terminals run with telnet 3270 or 5250 sessions. The mainframe could serve as the Domain Name Server (DNS).

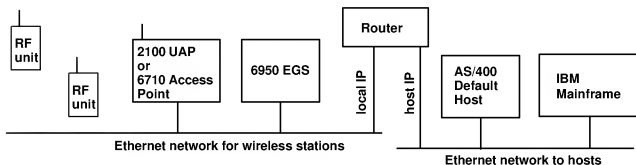


Figure 3-2
6950 EGS and Access Point Linked to Host Via Router

Connecting a Controller or Gateway

An RCB4030 Base/Controller, RC4030E Gateway, or RC3250 Network Controller can directly connect to the 6950 EGS. A controller such as the RCB4030 or RC3250 is only required when you need a UHF radio solution or do not want to attach the base radios to your Ethernet network.

RC3250 Network Controller

In Figure 3-3, an RC3250 Network Controller host port connects to COM1 on the 6950 EGS with a serial RS-232 connection. The controller sends and receives its radio messages via the RB3000 Base Station. The base radio communicates with UHF RF wireless stations.

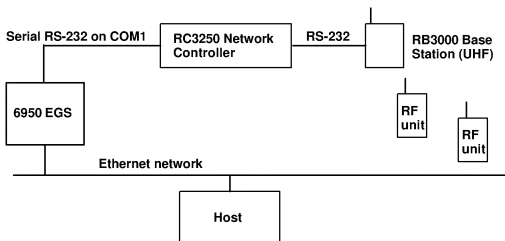


Figure 3-3

RC3250 Network Controller Connected to 6950 EGS

In Figure 3-4, the serial connection with the RC3250 Network Controller uses modems. The local modem attaches to COM1 on the 6950 EGS with a serial RS-232 connection. The base radio attaches to the network controller. The controller-to-base connection is RS-232 for UHF radios.

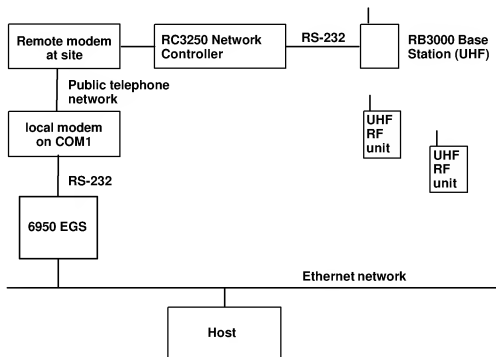


Figure 3-4
**RC3250 Network Controller Attached with Modem to
 6950 EGS**

You can use a modem with either type of serial connection and any base the controller supports. A modem configuration suits when the 6950 EGS is remote from the wireless network.

RCB4030 Base/Controller

In Figure 3-5, the 6950 EGS uses a serial connection to an RCB4030 Base/Controller. The RCB4030 Base/Controller communicates with the RB4030 Base Radio via radio. The controller sends and receive radio messages with SST RF wireless stations.

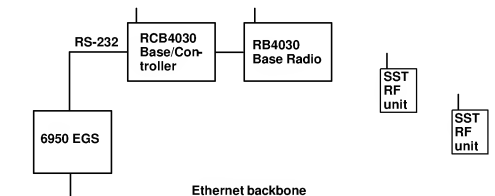


Figure 3-5
6950 EGS Connected to RCB4030 Base/Controller

RC4030E Gateway

In Figure 3-6, the 6950 EGS uses a serial connection to an RC4030E Gateway, which can communicate via Ethernet with a 6710 Access Point. Radio links the synthesized UHF RF wireless stations. The gateway could be removed, since the controller functions are integrated in the 6950 EGS.

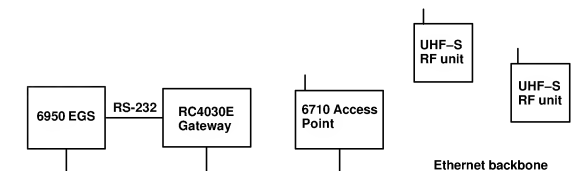


Figure 3-6
6950 EGS Connected to RC4030E Gateway

Front Panel

The 6950 EGS has two indicators on the front panel (Figure 3-7). The “Power” indicator lights when the 6950 EGS is plugged in. The “HD” indicator flashes when the hard disk drive is being accessed.

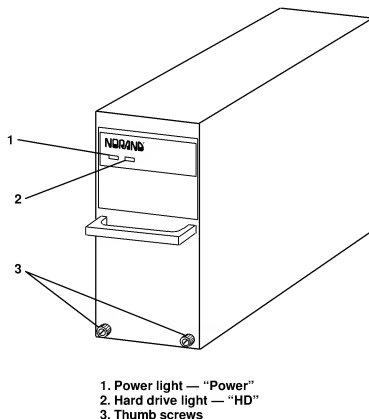


Figure 3-7
Front Panel

The front panel also has thumb screws for the mounting bracket.

Back Panel

The 6950 EGS has only one operating control, the power switch, on its back panel. Two models of the 6950 EGS have been produced. Both models have the following connectors:

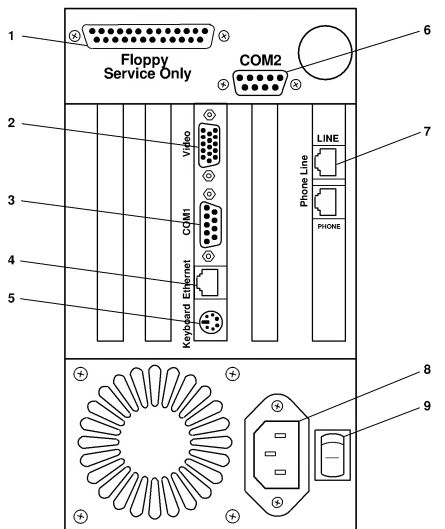
- ▶ 25-pin for external disk drive (updates only)
- ▶ 15-pin video monitor
- ▶ 2 9-pin serial communication ports, both configured for RS-232 communication
- ▶ RJ45 jack for Ethernet 10BASE-T
- ▶ PS/2 keyboard
- ▶ “LINE/PHONE” jacks to a V.32 and V.42 internal modem set to autoanswer for remote 6950 EGS configuration. (Modems are optional.)
- ▶ Switchable 110, 220, or 240 V ac power supply

Figure 3-8 shows the rear panel of the current model. Note that on the current model, the RJ45 jack is also the connector for 100BASE-T.

Early models of the 6950 EGS also have these components:

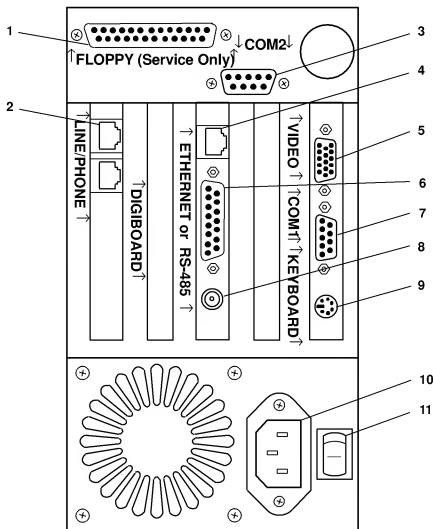
- ▶ 15-pin AUI port for Ethernet 10BASE5
- ▶ BNC connector for Ethernet 10BASE2

Figure 3-9 shows the rear panel of the early model.



1. External disk drive connector
2. Video connector (dumb terminal)
3. COM1 connector
4. RJ45 jack for 10BASE-T or 100BASE-T Ethernet connection
5. PS/2 keyboard connector (dumb terminal)
6. COM2 connector (dumb terminal)
7. Modem jacks
8. Power supply socket
9. Power switch

Figure 3-8
Back Panel of Current Model of 6950 EGS



1. External disk drive connector
2. Modem jacks
3. COM2 connector (dumb terminal)
4. RJ45 jack for 10BASE-T Ethernet connection
5. Video connector (dumb terminal)
6. AUI jack for 10BASE5 Ethernet connection
7. COM1 connector
8. BNC connector for 10BASE2 Ethernet connection
9. PS/2 keyboard connector (dumb terminal)
10. Power supply socket
11. Power switch

Figure 3-9
Back Panel of Early Model of 6950 EGS

During normal Ethernet operation, you do not use the floppy drive, video, keyboard, or either of the communication ports.

On the early model of the 6950 EGS, only one of the Ethernet connectors can be used at a time.

If configured for RS-232 communication, the 6950 EGS can use one communication port (COM1) to connect to an RC3250 Network Controller, RCB4030 Base/Controller, or RC4030E Gateway. The COM2 port is always reserved for dumb terminal operation and cannot operate RF equipment.

Location

Locate the 6950 EGS indoors within 7 running feet of an electrical outlet so the alternating current power cord reaches the power supply. Choose a location where you can see the LEDs on the front panel and reach the power switch on the back panel.

Mounting Brackets

The 6950 EGS can stand free on a counter or shelf, or you can mount it on a shelf or wall bracket. If you choose to use a bracket, you need one of three optional mounting brackets. The bracket part numbers (PN) follow.

PN	Description	Dimensions
850-529-102	Wall mount bracket	16.8 x 12.75 x 5.25 inches 42.7 x 32.4 x 13.3 cm
850-529-101	Shelf mount bracket for single 6950 EGS	16.8 x 5.4 x 3.75 inches 42.7 x 13.7 x 9.5 cm
850-529-103	Shelf mount bracket for two 6950 EGSs (double width)	16.8 x 9.8 x 3.75 inches 42.7 x 24.9 x 9.5 cm

To install a bracket for the 6950 EGS:

1. Use holes in the bracket as a template to mark wall or counter holes for mounting hardware (purchased locally).
2. Drill holes and install anchors (if used).
3. Position bracket and install mounting hardware.
4. Remove two thumb screws on the lower front panel of the 6950 EGS.
5. Slide the 6950 EGS into position.
6. Thread thumb screws through the 6950 EGS and mounting bracket.
7. Tighten thumb screws.
8. Attach cables.

Cabling

The cable required for the Ethernet connection is a standard device cable for your network. We sell the cable for connecting the RC3250 Network Controller's host port to the 6950 EGS's COM1 port. Its part number is 216-772-0xx, where "xx" indicates the cable length. Contact your representative for details.

Collecting the Equipment

Before you install the 6950 EGS, collect the equipment you need. The early model of the 6950 EGS directly connects to 10BASE2 or 10BASE-T Ethernet medium. The current model connects to 10BASE-T or 100BASE-T.

10BASE2 Components

10BASE2 components include the proper lengths of 10BASE2 coax cable, a T-connector, and a cable terminator. On the early model of the 6950 EGS, the 10BASE2 T-connector (Figure 3-10) attaches to the 6950 EGS's 10BASE2 port, and connects the 6950 EGS to the middle or end of 10BASE2 cable.



Figure 3-10
T-connector

A cable terminator (Figure 3-11) attaches to the T-connector. It is required for a device connected to the end of 10BASE2 cable. The terminator properly terminates the network cable to maintain proper impedance. Proper termination is necessary for reliable network communications.



Figure 3-11
Cable Terminator

10BASE-T or 100BASE-T Component

The 10BASE-T or 100BASE-T component is a cable that can extend up to 328 feet (100 meters) in length. The cable has an RJ45 plug at each end (Figure 3-12). Typically, the cable from the 6950 EGS to the RJ45 jack is less than 10 feet long.



Figure 3-12
Cable With RJ45 Plugs

Connecting to Ethernet

The following pages show how to connect the 6950 EGS to 10BASE-2 (early model only), 10BASE-T, and 100BASE-T (current model only).

10BASE2

The early model of the 6950 EGS can connect to the end or middle of the 10BASE2 cable segment.

► **NOTE:**

Cable lengths between network devices on the 10BASE2 Ethernet LAN must meet ANSI/IEEE standards.

End of Segment

See the following procedure and Figure 3-13.

1. Plug the T-connector (2) into the BNC port.
2. Plug one end of the Ethernet cable (3) into an open end of the T-connector. Align the notches in the cable end with the posts on the T-connector, push the cable in, and twist one-fourth turn.
3. Plug the cable terminator (1) into the other end of the T-connector.

Middle of Segment

See the following procedure and Figure 3-14.

1. Plug the T-connector (2) into the BNC port.
2. Plug one end of the Ethernet coaxial cable (1) into an open end of the T-connector. Align the notches in the cable end with the posts on the T-connector, push the cable in, and twist about one-fourth turn.
3. Plug the end of another Ethernet coaxial cable segment into the other open end of the T-connector.

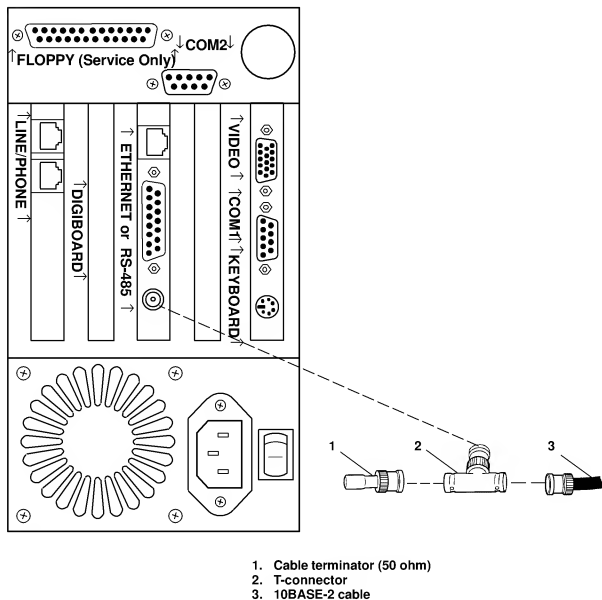


Figure 3-13
6950 EGS at End of 10BASE2 Segment

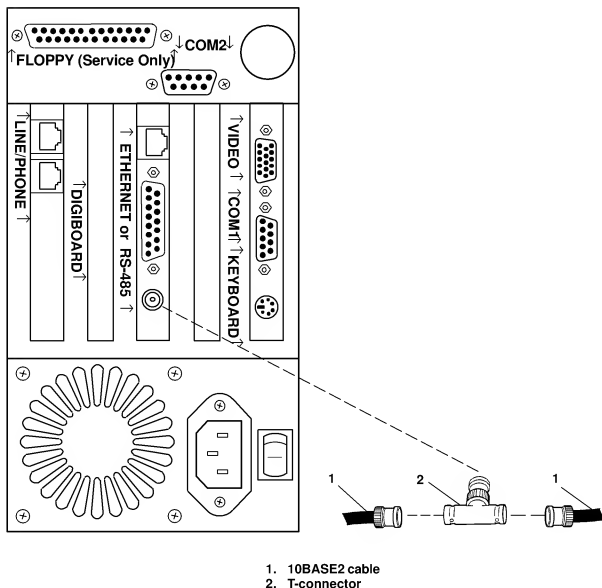
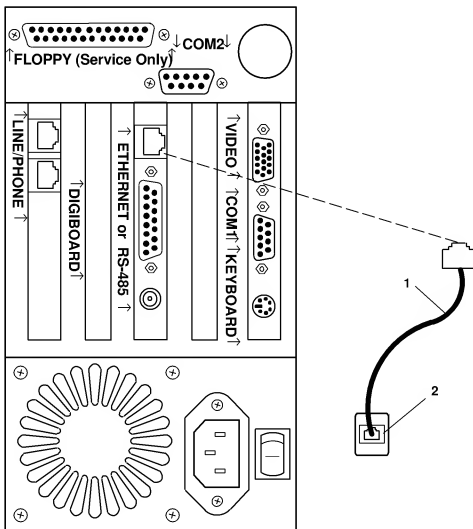


Figure 3-14
6950 EGS in Middle of 10BASE2 Segment

10BASE-T or 100BASE-T

See the following procedure and Figure 3-15.

1. Plug the cable with RJ45 jacks (1) into the 10BASE-T port.
2. Plug the other end of the cable into the RJ45 jack or hub port (2).



1. Cable with RJ45 plugs
2. RJ45 jack (or hub port)

Figure 3-15

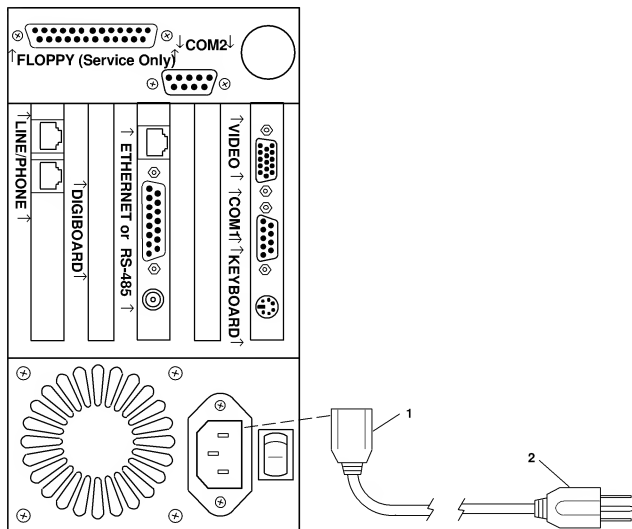
6950 EGS Connected to 10BASE-T or 100BASE-T

Applying Power

Power is applied to the 6950 EGS through the grounded ac input socket. To apply power, see Figure 3-16 and follow this procedure:

1. Plug the receptacle end of the power cord (1) into the ac input socket.
2. Insert the three-prong plug on the other end of the power cord (2) into a grounded power outlet.
3. Switch on the power. The “Power” indicator on the front panel should light.

If the 6950 EGS loses power during normal operation, it restarts operation when power returns. The power loss causes no harm to the 6950 EGS or the wireless network.



1. Receptacle on power cord
2. Three-prong plug

Figure 3-16
AC Power Connection

[illegible]

- ▶ Telnet session from another device on the network (see page 4-2)
- ▶ Video monitor and keyboard attached to the 6950 EGS (see page 4-4)
- ▶ Dumb terminal plugged into COM2 (see page 4-5)

Version 6.02 or greater of the wireless station's firmware provides VT220, TN3270, and TN5250 terminal emulation; and NORAND[®] Native using sockets. That means that once you set up the 6950 EGS with your network information, a wireless station can telnet to any application on a VAX, UNIX, or IBM host that uses one of these data streams.

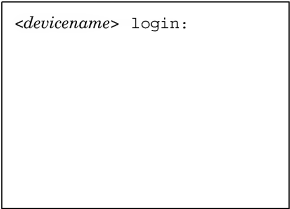
Telnet

1. Open a telnet session with the 6950 EGS. For initial configuration, you need the addresses set at the factory:

Default IP address: 198.70.37.28
Subnet mask: 255.255.255.0
Network address: 198.70.37.0

After initial configuration, use the IP address, subnet mask, and network address you set up for subsequent telnet sessions.

The login screen appears:



<devicename> login:

2. Enter this user name: **config**

The following screen appears:

```
<devicename> login:  
Password:
```

3. Enter this password: **10425rc** (password characters do not appear). The System Menu appears:

```
EGS Version <version>  
- - System Menu - -  
  
1 = Config  
2 = Control  
3 = Monitor  
4 = Update  
E = Exit  
  
Option->
```

The System Menu sets the conditions for the network.

4. Press:
 - 1 To configure network specific parameters such as IP addresses and host names, configure RF parameters and wireless stations, ping a host, open a telnet session, and set the date and time. See Section 5, “Configuring Network Options” and Section 6, “Configuring Radio Frequency Communication.”
 - 2 To halt or reboot the 6950 EGS, or restart without rebooting. See Section 7.
 - 3 To monitor the system, view log files on the 6950 EGS, and test the throughput of the RF link. See Section 8.
 - E To exit the System Menu.

► **NOTE:**

The ability to update the 6950 EGS software through option 4, “Update,” is currently unavailable.

Video Monitor and Keyboard

1. Attach the video cable to the connector marked “Video” and the keyboard to the “Keyboard” socket.
2. Begin a communication session. The login screen appears:

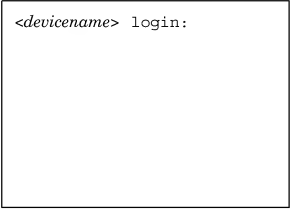


`<devicename> login:`

3. Follow the steps beginning with step 2 on page 4-2.

Dumb Terminal

1. Connect a dumb terminal (for example, a PC running a communication package) to COM2 with a null modem cable. Communication settings are:
 - ▶ Speed = 9600 bits per second
 - ▶ Parity = none
 - ▶ Data bits = 8
 - ▶ Stop bit = 1
2. Begin a communication session. The login screen appears:



```
<devicename> login:
```

3. Follow the steps beginning with step 2 on page 4-2.

Section 5

Configuring Network Options



This section describes how to set the network specific information the 6950 EGS uses to communicate with the network, including:

- ▶ IP addresses
- ▶ Host names
- ▶ Netmasks
- ▶ Domain Name Server (if used)

We have provided space for you to record this information as you set up the 6950 EGS. We encourage you to do so. If the network configuration is not successful, you have a convenient record for troubleshooting.

Required Information

You need the following information to configure the 6950 EGS. Contact your network administrator for assistance. You also need the information to update the 6950 EGS software. Names are case sensitive.

6950 EGS IP address _____ . _____ . _____ . _____

6950 EGS domain name _____

Network IP address _____ . _____ . _____ . _____

Subnet mask _____ . _____ . _____ . _____

Default host IP address _____ . _____ . _____ . _____

DNS host name (optional) _____

DNS IP address (optional) _____ . _____ . _____ . _____

Secondary

DNS IP address (optional) _____ . _____ . _____ . _____

(900 MHz RB4030 Base Radio system only) LAN ID _____

Wireless network host name _____

Static route:

Router local address (optional) _____ . _____ . _____ . _____
(if host is on the other side)

Router destination
address (optional) _____ . _____ . _____ . _____
(if host is on the other side)

Subnet mask (optional) _____ . _____ . _____ . _____

Default gateway IP address (optional) _____ . _____ . _____ . _____

Ethernet frame types:

Network devices must be enabled to pass the following DIX 2.0 Ethernet frame types (between the 6950 EGS and the access points only): 875a, 875b, and 875c. The network devices must also be enabled to pass multicast frames.

For help with setting frame types, refer to the network device's user manual.

Getting Started

When you log in to the 6950 EGS, you see the System Menu:

```
EGS Version <version>
- - System Menu - -

1 = Config
2 = Control
3 = Monitor
4 = Update
E = Exit

Option->
```

1. To configure network options, press “1” and then [Enter]. The Config Menu appears:

```
- - Config Menu- -

1 = IP
2 = RF Comm
3 = RF Units
4 = Date
E = Exit

Option->
```

2. Press:
 - 1 To use the IP Menu to configure and view your network specific parameters (for example, IP addresses and host names), ping a host, and open a telnet session. See “Using the IP Menu.”
 - 2 To configure RF communication parameters (for example, LAN ID [900 MHz RB4030 Base Radio systems only], host name, reconnect, automatic connect, and debug levels). See Section 6, “Configuring Radio Frequency Communication,” for details.
 - 3 To configure each wireless station; for example, its wireless station number, the host it logs in to, and the user ID it logs in as. Also use this option to copy one setup to other wireless stations. See “Setting Up the RF Units” later in this section.
 - 4 To set the date and time on the 6950 EGS. You need this information for log files that are date and time stamped. See “Setting the Date and Time” later in this section.
- E To exit the Config Menu.

Using the IP Menu

► **NOTE:** *Contact your network administrator for help with TCP/IP addresses, subnet masks, default gateways, and static routes. If one of these items is incorrect, communication between the 6950 EGS and host computer will not work.*

1. At the Config Menu, press “1.” The IP Menu appears:


```
IP Menu
1 = View
2 = Config
3 = Ping
4 = Telnet
E = Exit
->
```

2. Because you must configure IP addresses before option 1, “View” and option 3, “Ping” are useful, press “2” for the Config Menu and then [Enter]. The IP Config menu appears:

```
- IP Config -
1 = IP
2 = Hosts
E = Exit
```

```
Option->
```

3. Press:
 - 1 To configure IP addresses. See “Configuring IP Addresses.”
 - 2 To add or delete host names and matching IP addresses. See “Configuring Host Names” later in this section.
 - E To exit.

Configuring IP Addresses

► **NOTE:** Before you configure IP settings, ensure that you have the IP information listed on page 5-2.

When you press “1” at the IP Config menu, the following screen appears:

```
Norand EGS
Configuration
Version <version>

1. Enter the
hostname for
this EGS:
->
```

<version> represents the 6950 EGS's current version.

To configure IP settings:

Complete the configuration steps on the following pages. If you make a mistake, you can press <Ctrl> <C> to start over.

The first two steps construct a name for this 6950 EGS that can be used by a DNS. It takes the form <hostname> <domain>. If you do not use a DNS, you must still type something for each request; for example use “Intermec” for host, and “radios” for domain. Your network uses the IP addresses, supplied following the names.

1. Type the host name for this 6950 EGS (for example, “Intermec”) and press [Enter]. The host name may be up to 8 characters long. Alphanumeric characters are acceptable in upper or lowercase. The underscore is acceptable, but spaces are not.

► **NOTE:**

Do not use this host name for your wireless stations. Their host name identifies a radio controller (such as the 3250 Network Controller or RCB4030 Base/Controller). Do not confuse this IP host name with the name of the host to which the wireless stations telnet. The host name requested here is for the 6950 EGS.

The following screen appears:

```
2. Enter domain
name for <hostname>
without the '.':
->
```

<hostname> is the name from the previous step. The domain name sets the domain within the host network that the 6950 EGS can reach. For initial testing, use this default name: NORAND

The period “.” referred to in this prompt is the period that falls between host name and domain in the full address. Do not begin the domain name with a period. For example, use “company.com” instead of “.company.com” to construct the name “hostname.company.com”.

2. Type the domain name and then press [Enter]. The following screen appears:

```
Unit name is  
<hostname>.<domain>
```

```
3. Enter IP  
address for this  
EGS:  
->
```

3. Type the IP address assigned to this 6950 EGS by your network administrator (for example, "111.111.111.111") and press [Enter]. The following screen appears:

```
4. Enter IP  
address for  
this network:  
->
```

4. Type your network IP address (for example, "111.111.111.0") and press [Enter]. The following screen appears:

```
5. Enter network
netmask:
->
```

A subnet mask takes the form ###.###.###.###, where each “###” segment represents a decimal address from 0–255.

5. Type your network mask (subnet mask) (for example, “255.255.255.0”) and then press [Enter]. The following screen appears:

```
6. Enter default
host IP address:
->
```

Your default host is the first host the 6950 EGS attempts to connect with on wireless station start-up. When the 6950 EGS detects a wireless station powering on, it tries to open a telnet connection to this host. For example, your most commonly used application program might reside there. Choose the first host you want the 6950 EGS to connect with when a wireless station powers on.

6. Type your default host IP address (for example, “111.111.111.5”) and press [Enter].

The next screen appears:

```
7. Use default
gateway? Y/N
->
```

If you have a specific gateway you always want the 6950 EGS to attempt its first connection through, use a default gateway. If the 6950 EGS fails to connect to the default gateway, it tries other routes. Consider Figure 5-1.

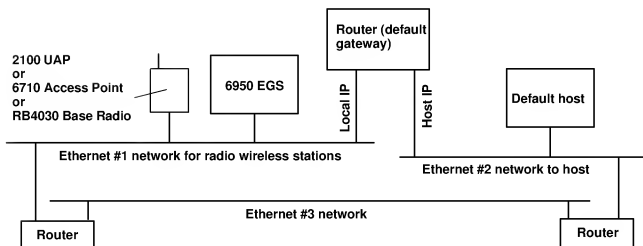


Figure 5-1
Default Gateway to Host

In this example, the 6950 EGS can reach the default host by two different routes:

- ▶ Through the default gateway
- ▶ Through Ethernet network #3

If you use a default gateway, the 6950 EGS always attempts first to reach the default host through the default gateway.

If an attempt to reach the default gateway fails, the 6950 EGS attempts any other available path. In the example, the alternate route passes along Ethernet network #3.

7. If you are not using a default gateway, press “N.” The next screen asks if you want to use a static route. Skip to step 8.

If you are using a default gateway, press “Y” and then [Enter]. The following screen appears:

```
Enter
Default
Gateway:
->
```

Type the default gateway address and press [Enter]. This address too takes the form “111.111.4.3.” Be sure to use the IP address for the wireless network side of the gateway; “local IP” as shown in Figure 5-1.

The following screen appears:

```
8. Use static
route? (Y/N)
->
```

Reserve static routes only for a single fixed path from the 6950 EGS to a larger network. A static route is a single, fixed path between the 6950 EGS and the host computer. Static routes keep the network from attempting other routes if the static route is unsuccessful or fails during operation. Typically a router must pass along all traffic for an otherwise isolated wireless network. See the example in Figure 5-2.

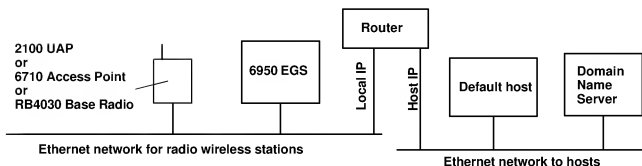


Figure 5-2
Static Route to Host

8. If you are not setting a static route, press “N.” You are asked if you want to use a DNS. Skip to step 9. If you are using a static route, press “Y” and then [Enter] to set a new static route for the 6950 EGS to use. The following screen appears:

```
Enter
Destination
Address:
->
```

Type your destination address. This is the network you are reaching through the router (for example “198.70.37.0”).

Press [Enter]. The following screen appears:

```
Enter Gateway
Address:
->
```

Type the gateway address for the static route. This is usually the address of the router (for example, “136.179.76.3”). Be sure to use the IP address on the side of the router with the RF equipment. In Figure 5-2 on page 5-12, the connection is marked “local IP.”

Press [Enter] after typing the address. The following screen appears:

```
Enter Static  
Route Netmask:  
->
```

Type the subnet mask for the static route and press [Enter]. The subnet mask must give the 6950 EGS access to the default host. This completes the static route.

► **NOTE:**

After you complete setup and reboot the 6950 EGS, you can check the static route by pinging the host from the 6950 EGS. The host can check the return path by pinging the 6950 EGS.

The following screen appears after you enter the netmask:

```
9. Use Domain  
Name Server?  
Y/N  
->
```

A DNS is a specially configured computer that does name resolution. If you type a host name, it translates that into the IP address of that host.

9. If you are not using a DNS, press “N” and then [Enter]. Skip to step12.

To use a DNS, press “Y” and then [Enter]. The following screen appears:

```
NAMESERVER setup for:  
<IP> <hostname> <domain>  
  
10. Enter IP  
address of the  
Name Server for  
domain <domain>:  
->
```

10. Type the DNS host IP address and press [Enter]. The following screen appears:

```
NAMESERVER setup for:
<IP> <hostname> <domain>
```

```
11. IP Address of
Secondary Domain
Name Server:
<ENTER if none>
->
```

11. If you are using a secondary (fallback) DNS, enter its IP address. If you are not using one, press [Enter].
12. The following message confirms the configuration was saved and is complete:

```
Configuration is
complete.
```

```
Restart for
changes to take
effect.
```

```
Press any key:
```

13. Press [Enter] to restart the 6950 EGS. The following screen appears:

```
Are you sure
you want to
reboot?  (y/n)
->
```

You must reboot the 6950 EGS before the changes you made can take effect.

14. Do one of the following:
 - ▶ To not reboot, press “N” and [Enter]. The Config Menu returns. The 6950 EGS continues to operate but without the configuration changes taking effect. They do not take effect until you reboot the 6950 EGS.
 - ▶ To reboot, press “Y” and then [Enter]. It takes about half a minute to a minute for the 6950 EGS to reboot.

Allow about 3 minutes for the wireless network to restart. Messages like the following build one after the other on the display as the restart occurs:

```
Reboot in  
Progress. . .  
Terminal will  
stop.  
Wait for reconnect  
prompt.  
  
Stopping RF.
```

When the reboot is complete, the wireless station beeps and the message “RF Server Ready!” returns. The 6950 EGS is now configured and ready for regular operation. If the message does not appear, the wireless station may be out of range of the 2100 UAP, 6710 Access Point, or RB4030 Base Radio.

Configuring Host Names

Use the IP addresses and host names supplied with option 2 on the IP Config menu to build a local host table.

Wireless station users can then supply the host names instead of IP addresses during connection.

Local host tables are useful when a DNS is not present on the Ethernet network. A name is less cumbersome and prone to fewer errors than an IP address.

The 6950 EGS keeps a table of each host name and IP pair. When users supply a host name, it substitutes the IP address. For example, assume the host name "IRV" has been added with the IP address 192.168.5.40. When users supply the host name "IRV," the 6950 EGS translates that to address 192.168.5.40 and opens a connection to the host.

The 6950 EGS checks its local host table for IP addresses before checking the DNS.

To configure host names:

1. Press "2" at the IP Config menu. The HOST Menu appears:

```
HOST Menu
1 - Add
2 - Delete
E - Exit

Option ->
```

2. Press:
 - 1 To add a host by supplying the IP address. See “Adding a Host.”
 - 2 To delete a host by supplying the IP address. See “Deleting a Host.”
 - E To exit the menu.

Adding a Host

1. At the HOST Menu, press “1.” The following screen appears:

```
Enter IP  
to add  
->
```

2. Enter the IP address and then press [Enter]. The following screen appears:

```
Enter Hostname  
to add  
->
```

3. Type the host name and then press [Enter]. You return to the HOST Menu.

Deleting a Host

1. At the HOST Menu, press “2.” The following screen appears:

```
Enter IP  
or Hostname  
to delete  
->
```

2. Type the IP address or host name you want to delete and press [Enter]. You return to the HOST Menu.

Viewing IP Addresses and Host Names

1. At the IP Menu, press “1” and then [Enter]. The IP View menu appears:

```
IP View  
1 = IP  
2 = Hosts  
E = Exit  
->
```

2. Press “1” and then [Enter] to view the current configuration. For example:

```
IP Address = <IP address>
Netmask = <netmask>
Network address = <IP address>
Static Route:
  Gateway =
  Destination =
  Netmask =
Default Gateway = <IP address>
Domain <domain>
Name Server <IP DNS>
Name Server <secondary DNS IP>
Press <Enter>
```

If you configured a static route or a DNS, its settings appear on this screen.

3. When finished, press [Enter].
4. At the IP View menu, press “2” to view the host set up. The IP address and host name for each host appear. For example:

```
198.70.37.30 test
198.70.37.31 new
198.70.37.32 Lynda
198.70.37.33 Joe
198.70.37.34 D_Doehr
Press <Enter>
```

5. When finished, press [Enter].

Pinging a Host

► **NOTE:**

The ping feature will not function until you complete the IP network configuration and reboot the 6950 EGS. For instructions, see "Using the IP Menu" earlier in this section.

To ping a host:

1. At the IP Menu, press "3." The following screen appears:

```
Enter hostname
or press
<ENTER> for
default host

HOST->
```

2. Type the IP address or host name and then press [Enter].

If the response is successful, the response from the host appears and the 6950 EGS can communicate with the specified host. If the ping fails, review the network connections and configurations.

3. Press [Enter] to continue.

Opening a Telnet Session

The wireless station can telnet to any host that supports the VT/ANSI, TN3270, or TN5250 data stream.

To open a telnet session:

1. At the IP Menu, press “4” and then [Enter]. The following screen appears:

```
Enter hostname
or press
<ENTER> for
default host

HOST->
```

2. Type the host name or IP address to telnet to and press [Enter].

The 6950 EGS attempts to connect to the host (it shows the IP address it is trying). If the host does not respond because of a faulty connection path, is not prepared for telnet, or is down, the 6950 EGS stops the attempt and suggests possible reasons.

If you do not enter a name or an IP address, the default for the 6950 EGS is to connect to the host defined as the default host using port 23 (localhost=default host).

Setting Up the RF Units

Each wireless station is assigned a host to which it will connect. This assignment is made in the 6950 EGS. A network may consist of several access points, several 6950 EGS gateways, and several host computers. One 6950 EGS can assign wireless stations to multiple hosts. Each wireless station must be configured to have a unique terminal number and host name combination. Wireless stations may have the same terminal number, but they must then be assigned to different 6950 EGS gateways with different host names.

To set up the wireless station, press “3” at the Config Menu (shown on page 5-3). The following screen appears:

```
nui_cfg Wireless Network Access Server Configuration -- Modlvl <version>

RF Unit Setup:

1. Unit number .: 1

3. Comment.....: VT220 terminal

5. IP Host name.: localhost
6. Port.....: 0023
7. User ID.....:
8. Password.....:
9. Copy Setup from unit ???
10. Duplicate this setup to unit ??? thru unit ???

CTRL->Enter option: (E=Exit,N=next,P=previous or item number to change)
HELP->
```

The following chart describes the options.

Option	Description
5. IP Host name	Default host to which the wireless station attaches. "Localhost" is the default host configured in the IP configuration.
6. Port	TCP port number the 6950 EGS tries to connect to when creating the telnet connection. Port 23 is the "well-known" standard telnet port. To connect to a different TCP port, supply the 4-digit number here.
7. User ID	<p>If you supply this option and if you are using VT220 terminal emulation (without telnet), the 6950 EGS attempts a remote login to the IP host defined with User ID. Autologin works only if the end of the host prompts match any of the following. <i>For the autologin prompt from your telnet host:</i></p> <p>*sername; *SERNAME; *ogin; *OGIN;</p>

Option	Description
8. Password	<p>Password supplied to the host when trying to connect. Autologin works only if the end of the host prompts match any of the following. <i>For the password prompt from your telnet host or from another unit to this unit:</i></p> <p>*ASSWORD: *assword:</p>
9. Copy Setup from unit ??? and	Enable you to copy one setup to a group of other wireless stations in
10. Duplicate this setup to unit ?? thru unit ???	one step.

To configure the wireless stations:

1. Type the unit number (option 1) and the IP host name (option 5). Add a comment (option 3) if desired. Press [Enter].
2. To add remaining wireless stations, use one of these methods:
 - For a single terminal, type the number from the desired terminal's setup at line 9.
 - To copy the current set up to several terminals, type the range of terminal numbers at line 10.
3. Press [Enter].

Setting the Date and Time

1. At the Config Menu, press “4.” The Date Menu appears:

```
-- Date Menu --  
<date> <time>  
  
1 = Change  
E = Exit  
Option ->
```

2. Press “1” to change the date or time:

```
Current Date:  
<date>  
New date:  
<MM/DD>
```


3. To change the date, type a new date on the blank line in the same form as the current one. Use a one- or two-digit month (MM) and a one- or two-digit day of the month (DD), supplying a slash.

To leave the date unchanged, press [Enter] and check the time:

Current Date:

<date>

New date:

<MM/DD>

Current Time:

<time>

New Time:

HH:MM

4. To change the time, type a new time (24-hour clock) on the blank line in the same form as the current one. Use a one- or two-digit hour (HH) and two-digit minutes (MM), supplying a colon between them. To leave the time unchanged, press [Enter] to return to the Date Menu.

[illegible]

To begin the configuration:

1. Start at the System Menu (for directions to the menu see Section 4, *Starting the 6950 EGS*):

```
Version <version>
-- System Menu --

1 = Config
2 = Control
3 = Monitor
4 = Update
E = Exit

Option->
```

2. Press “1” and then [Enter]. The Config Menu appears:

```
-- Config Menu --  
  
1 = IP  
2 = RF Comm  
3 = RF Units  
4 = Date  
E = Exit  
  
Option->
```

3. To configure the wireless network through Ethernet, see “Configuring Through an Ethernet Connection.” To configure through RS-232, see “Configuring Through an RS-232 Connection” later in this section.

Configuring Through an Ethernet Connection

Press “2” at the Config Menu. The Comm Setup menu appears:

```
Comm Setup  
  
1. Type: E (E,R)  
2. Lan : <LAN ID> (0-7)  
3. Host: <host name>  
4. Rec : 0 (0,1)  
5. Auto: 0 (0,1)  
6. Dbg : 0 (0-2)  
Option: (1-6,E)
```

The screen lists your current RF configuration. Characters within parenthesis show choices for the numbered option.

► **NOTE:** *If you are using a 900 MHz RB4030 Base Radio and change the LAN ID for the 6950 EGS, you must change the LAN ID for the wireless stations and the base radio to match before the wireless station lets you log in again.*

► **NOTE:** *If you change the host name for the 6950 EGS, you must change the host name for the wireless stations to match before the wireless station lets you log in again.*

The following chart describes the options.

Option	Name	Function
1	Type	Displays the type of network setting: E = Ethernet (<i>default</i>) R = RS-232 serial communication
2	Lan	Displays the LAN ID. The default is 0 (zero). If you are using a 2100 UAP or 6710 Access Point, use LAN ID 0 for the 6950 EGS regardless of the access point's LAN ID setting and type of radio (802.11, 2.4 GHz OpenAir, 900 MHz, or synthesized UHF). If you are using a 900 MHz RB4030 Base Radio, the 6950 EGS's LAN ID must match the base radio's LAN ID. ► NOTE: <i>The wireless stations' LAN ID must match the LAN ID for the 2100 UAP, 6710 Access Point, or RB4030 Base Radio regardless of the type of radio in these devices.</i>
3	Host	Displays the wireless network's host (controller) name. The default is "HOST." ► NOTE: <i>The host name is case sensitive. The wireless stations must be configured to match this host name.</i>

Option	Name	Function
4	Rec (Non-telnet VT220 terminal emulation only)	<i>Reconnect</i> . Controls whether the 6950 EGS reattaches to the previous session after a wireless station has been powered off then on. NOTE: <i>The wireless station's display does not refresh after reconnection.</i> Settings: 0 = off (default) 1 = on
5	Auto (Non-telnet VT220 terminal emulation only)	<i>Automatic login</i> : Controls whether the 6950 EGS automatically attempts to telnet to the host defined for RF Unit Setup (described on page 5-25). Settings: 0 = off (default) 1 = on
6	Dbg	<i>Debug</i> : Verbosity level (detail kept in the log file). See details for the log file in Section 8, "Monitoring the System." Settings: 0 = silent (default) — Reports no debug information during normal operation. Leave Debug in this mode unless otherwise instructed by our personnel. 1 = normal — Messages between 6950 EGS and wireless stations about processes running. 2 = debug — Debug information about processes named during "normal" mode. ► NOTE: <i>Under normal conditions use debug level zero, since the additional activity at the other two log levels degrades performance.</i>

To configure the options:

1. Press the number next to the item you must change, then make the change. Choices are shown in parenthesis.

If you do not want to make any changes, press “E” to return to the Config Menu.

2. If you made any changes, you are asked to save them after you press “E” to exit. To discard the changes, press “N” (screen not shown here).

To save the changes, press “Y.” The following screen appears:

```
!!! WARNING !!!  
Lan=<lan> Host=<host>  
Press the = key  
if you are very  
sure  
->
```

3. Write down the LAN ID and host name listed here. They appear as <lan> and <host> respectively.

► NOTE:

If you do not write down this information and your wireless station does not let you log back in, you must log in by another method to learn these values. For example, with the IP address you could begin a telnet session.

4. Press “=” once you have the information. A second message tells you the configuration was saved (not shown here). Press [Enter].
5. You must reboot or restart the 6950 EGS before these changes take effect. See Section 7, *Halting, Rebooting, or Restarting the 6950 EGS*, for details.

Configuring Through an RS-232 Connection

You must use one of the following methods for the initial configuration:

- ▶ Keyboard and monitor attached to the 6950 EGS
- ▶ PC attached to COM2 on the 6950 EGS
- ▶ Modem

You must set the following parameters on the RCB4030 Base/Controller or RC3250 Network Controller, or they will not work with the 6950 EGS.

- ▶ HOST = ASYNC
- ▶ PARITY = NONE
- ▶ STOP BITS = 1
- ▶ DATA BITS = 8
- ▶ MULTIPLE BUFFERING ENABLED
- ▶ CHAINING DISABLED
- ▶ POLLING IS REGULAR

To begin, press “2” at the Config Menu. The Comm Setup menu appears:

Comm Setup

1. Type: E (E,R)
2. Lan : <lan> (0-7)
3. Host: <host>
4. Rec : 0 (0,1)
5. Auto: 0 (0,1)
6. Dbg : 0 (0-2)
- Option: (1-6,E)

Press “1” to change the type and then “R” to change to RS-232. The screen changes to the following:

Comm Setup

1. Type: R (E,R)
2. Config RS232
4. Rec: 0 (0,1)
5. Auto: 0 (0,1)
6. Dbg: 0 (0-2)
- Option: (1-6,E)

The following chart describes the options.

Option	Name	Function
1	Type	Displays the type of network setting: E = Ethernet (<i>default</i>) R = RS-232 serial communication
4	Rec (Non-Telnet VT220 terminal emulation only)	<i>Reconnect</i> : Controls whether the 6950 EGS reattaches to the previous session after a wireless station has been powered off then on. NOTE: <i>The wireless station's display does not refresh after reconnection.</i> Settings: 0 = off (<i>default</i>) 1 = on
5	Auto (Non-Telnet VT220 terminal emulation only)	<i>Automatic login</i> : Controls whether the 6950 EGS automatically attempts to telnet to the host defined for RF Unit Setup (described on page 5-25). Settings: 0 = off (<i>default</i>) 1 = on
6	Dbg	<i>Debug</i> : Verbosity level (detail kept in the log file). See details for the log file in Section 8, "Monitoring the System." Settings: 0 = silent (<i>default</i>) — Reports no debug information during normal operation. Leave Debug in this mode unless otherwise instructed by our personnel. 1 = normal — Shows wireless station power-up and connection information. 2 = debug — Shows detailed messages in and out of the 6950 EGS, plus information contained in "normal" mode. ► NOTE: <i>Under normal conditions use debug level zero, since the additional activity at the other two log levels degrades performance.</i>

To configure the options:

1. Select the number next to the items you must change, then make the changes. Choices for each numbered option appear in parenthesis.
2. Press “2” (for Config RS232) and then [Enter]. The RS-232 Setup menu appears:

[illegible]

The rows of “Ys” and “Ns” below option 4 show which wireless stations are enabled: “Y” for yes and “N” for no. The first row shows wireless stations 00–63.

The following chart describes the options.

► **NOTE:**

You can change options 1–4 in any order.

Press	To
1	<p>Set the baud rate to match that of the RCB4030 Base/Controller or 3250 Network Controller. Valid options are:</p> <p>1 = 1200 bps 2 = 2400 bps 3 = 4800 bps 4 = 9600 bps 5 = 19200 bps (<i>default</i>) 6 = 38400 bps</p> <p>Press the digit preceding your communication speed and then press [Enter].</p>
2 (<i>United States only</i>)	<p>Set your FCC call sign (up to 10 characters) if you are transmitting with UHF radio frequency, and then press [Enter].</p>
3	<p>Enable a range of wireless stations. Press “3” and then type the wireless station numbers in the underlined area to enable a range of wireless stations at once. Press [Enter] after each number in the range. The 6950 EGS asks you to confirm the change. Press “Y” or “N”. Your changes appear in the rows of wireless station numbers.</p>
4	<p>Disable a range of wireless stations. Press “4” and then type the wireless station numbers in the underlined area to disable a range of wireless stations at once. Press [Enter] after each number in the range. The 6950 EGS asks you to confirm the change. Press “Y” or “N.”</p>
E	<p>Exit this menu and return to the Comm Setup menu.</p>

3. If you made any changes, you are asked to save them after you press “E” to exit. To save the changes, press “Y.” To discard the changes, press “N” (screen not shown here).
4. You must reboot or restart the 6950 EGS for the changes to take effect. See Section 7, *Halting, Rebooting, or Restarting the 6950 EGS*, for details.

Halting, Rebooting, or Restarting the 6950 EGS

To halt, reboot, or restart the 6950 EGS:

1. Begin at the System Menu (see Section 4, *Starting the 6950 EGS*, for instructions on how to access the System Menu):

Option->

2. Press “2” and then [Enter]. The Control Menu appears:

```
- Control Menu -  
  
1 = Halt  
2 = Reboot  
3 = Restart  
E = Exit  
  
Option->
```

3. Press:
 - 1 To halt the 6950 EGS. Halt stops all activity between the host application and the wireless stations, but does not reinitialize the 6950 EGS. Use this option if you need to turn off the 6950 EGS or move it. See “Halting the 6950 EGS.”
 - 2 To reboot the 6950 EGS. Reboot reinitializes the 6950 EGS, but does not reset its configuration settings. Reboot is equivalent with the <Ctrl>+<Alt>+ warm boot of a DOS personal computer. See “Rebooting the 6950 EGS” later in this section.
 - 3 To restart the software communicating with wireless equipment. See “Restarting the 6950 EGS” later in this section.
 - E To exit.

Halting the 6950 EGS

To halt the 6950 EGS, you can simply power it off or you can use the Control Menu. For information about the power switch and applying power, see Section 3, “Installation.” To use the Control Menu, see the following instructions.

1. At the Control Menu, press “1” and then [Enter]. This message appears:

```
Are you sure  
you want to  
shutdown?  
(y/n)  
->
```

2. To cancel shutdown, press “N” and then [Enter].

To shut down the 6950 EGS, press “Y” and then [Enter]. Please wait three minutes after selecting this option before powering the 6950 EGS off.

The following messages appear as the 6950 EGS shuts down. They are the last responses from the wireless station:

```
Shutdown in  
Progress...  
Wait 3 minutes.  
Terminal will  
stop.  
  
Stopping RF.
```

3. Restart the 6950 EGS by switching its power off and on.
4. Power the wireless station off and on. It takes about 3 minutes before the terminal shows its first screen.
5. When you see the message “RF Server Ready!” (VT220 wireless station) or the host login (TN3270 or TN5250 wireless station), the 6950 EGS and terminals have returned to normal operation.
For information about logging in to the host, see “Logging in to the Host” later in this section.

Rebooting the 6950 EGS

1. At the Control Menu, press “2” and then [Enter]. The following message appears:

```
Are you sure  
you want to  
reboot? (y/n)  
->
```

2. Press “Y” and then [Enter] to confirm you want to reboot. The following messages appear:

```
Reboot in  
Progress...  
Terminal will  
stop.  
Wait for connect  
prompt.
```

These messages show progress during shutdown. The final message, “Stopping RF,” shows completion of shutdown. The wireless station does not respond further. It takes about 3 minutes for the 6950 EGS to respond to the wireless station after it starts back up.

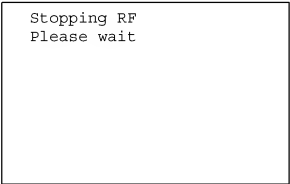
3. The wireless station beeps. When you see the message “RF Server Ready!” (VT220 wireless station) or the host login (TN3270 or TN5250 wireless station), the 6950 EGS and terminals have returned to normal operation.

For information about logging in to the host, see “Logging in to the Host” later in this section.

Restarting the 6950 EGS

1. At the Control Menu, press “3” and then [Enter]. It takes about three to five minutes to completely restart before the wireless stations receive their initial screen.

The following messages appear:



```
Stopping RF  
Please wait
```

These messages show when the 6950 EGS shuts down. They are the last responses from the wireless station until the 6950 EGS restarts.

2. When you see the message “RF Server Ready!” (VT220 wireless station) or the host login (TN3270 or TN5250 wireless station), the 6950 EGS and terminals have returned to normal operation. For information about logging in to the host, see “Logging in to the Host” later in this section.

Logging in to the Host

Log in to the host after your wireless station is configured with the proper LAN ID, host name, and terminal emulation type. Note that for Ethernet systems, wireless stations with 3270 or 5250 terminal emulation must have telnet enabled.

Turn on the wireless station. If it is already on, turn it off then on. The host login screens that appear vary for the type of terminal emulation.

VT220 Terminal Emulation

1. When you turn on the VT220 wireless station, the following screen appears:

```
RF Server Ready!  
  
Press '1' now to  
connect.  
->
```

2. Press “1” and then [Enter] to connect. The following screen appears:

```
Norand Corp.  
Rf Terminal  
Server V<version>  
      (0003)
```

```
Enter telnet host  
or press enter  
for (localhost).  
->
```

“Localhost” is the default IP host to connect to. *<version>* is the 6950 EGS version number. The number in parentheses is your wireless station's number.

3. To log in to the default host, press [Enter].
You can also type the IP address and press [Enter].
You can use the host name if you have a DNS and the IP address has a domain name in the 6950 EGS host table.

The following screen appears:

```
Trying <device>...  
press 'x' to  
stop trying.
```

<device> represents the host you have chosen.

4. You can press “x” to cancel the attempt. It will time out on its own if a connection could not be made. You are told if a connection could not be made and must press [Enter].

If you do not cancel the attempt, the following screen appears (this screen represents your host login):

```
<devicename> login:  
Password:
```

<devicename> shows the host you have logged in to.

5. Enter your login and password, and then press [Enter] after each. The first screen of your host application appears.

TN3270 or TN5250 Terminal Emulation

The first login screen that appears for a TN3270 or TN5250 wireless station is the default host login screen (shown on this page). Enter your login and password, and then press [Enter] after each. The first screen of your host application appears.

[illegible]

To monitor the system:

1. Begin at the System Menu (see Section 4, 6950 EGS, for instructions on how to access the System Menu):

```
- - System Menu - -  
  
1 = Config  
2 = Control  
3 = Monitor  
4 = Update  
E = Exit  
  
Option->
```

2. To monitor the system or view the log files on the 6950 EGS, press “3” and then [Enter].

The Monitor Menu appears:

```
- Monitor Menu -  
  
1 = System  
2 = Logs  
3 = Time Test  
E = Exit  
  
Option->
```

3. Press:

- 1 To see the wireless processes that are currently running. See "System Monitor."
- 2 To see the log of activity for the 6950 EGS and its interaction with the wireless network. See "Logs" later in this section.
- 3 To test the throughput of the RF link. See "Time Test" later in this section.
- E To exit and return to the System Menu.

System Monitor

The System Monitor varies for Ethernet and RS-232 connections.

To view the monitor:

1. At the Monitor Menu, press “1” and then [Enter] to view the wireless processes. Following is an example of the processes for an RS-232 connection:

```
368 p 1 S N 0:00 sh ./nui.sh
374 p 1 S N 0:00 ./nui_serv
375 p 1 S N 0:00 ./nui_comm
```

Following is an example of the processes for an Ethernet connection:

```
40 con S 0:00 ./nui_eth /dev/eth0
42 con S 0:00 sh ./nui.sh
47 con S 0:00 ./nui_serv
52 con S 0:00 ./nui_pps
66 con S 0:00 nui_mgr HOST 0
70 con S 0:00 ./nui_pps
152 p 1 s 0:01 sh /nuiscripts /main
513 p 1 s 0:00 /nui/nuigrep nui
```

Processes and activities appear on the right following the numbers of the form “0:00.” The numbers indicate CPU utilization.

2. Press [Enter] when finished.

Logs

Use the log files to view the activity for the 6950 EGS and its interaction with the wireless network. You can also view new information added to the logs, or clear the log file and then restart it.

For clarity, log messages show as they appear on a 24-line by 80-column monitor. The same information appears on a wireless station display, but you must scroll to read it.

The debug level controls the amount of information reported in the logs. For more information about setting debug levels, see page 6-4 (Ethernet connection) or 6-8 (RS-232 connection). For help with deciding which level of debug information you need, see “RS-232 Log Debug Levels” or “Ethernet Log Debug Levels” later in this section.

Each combination of access points or base radios, controllers, and wireless stations produces a different log. Your log may vary a lot and still be normal for your configuration.

To view, watch, or clear the log file:

1. At the Monitor Menu, press “2” and then [Enter]. The Log file menu appears:

```
Log file
1 - View
2 - Watch
3 - Clear & Restart
E - Exit

Option->
```

2. Press:
 - 1 To view the log. See “Viewing the Log File.”
 - 2 View new information added to the log. See “Watching the Log File” later in this section.
 - 3 Clear the log file and then restart it. See “Clearing the Log File” later in this section.
- E To exit.

Viewing the Log File

1. At the Log file menu, press “1” and then [Enter]. The server log appears. Following is a sample server log for an Ethernet connection:

```
<date><time> nuiserv> WNAS -- Version <version>
<date><time> nuiserv> Copr. 1991-1999 Intermecc. All rights reserved.
<date><time> nuiserv> System ID: LINUX norand 2.0.35 #15 <day><date><time> CDT 1999 i486
<date><time> nuiserv> Initialization in progress.
<date><time> nuiserv> Server files are in directory ./
<date><time> nuiserv> This is server number 0. Type E
<date><time> nuiserv> .. Opening message queue ...
<date><time> nuiserv> .. Spawning tty process...
<date><time> nuiserv> Child 000056: Exec 127, ./nui_pps. Type 2
<date><time> nuiserv> Server initialization complete.
<date><time> C000056> nui_func Modlvl 7.00: Parent process ID is 51.
<date><time> C000056> IPC OK! Server=0,Term=127,ID=2,Qid=0,rf lan=0,lancon=E
<date><time> C000056> nui_pps: Portable Protocol Stack Interface Modlvl 7.00
<date><time> C000056> Control channel created.
<date><time> C000056> .. Starting protocol stack daemons. RF Host HOST:0 io=0
<date><time> C000056> .. Starting nui_mgr ...
<date><time> C000056> Stack manager is started. Pid = 80
<date><time> C000056> PPS IO channel created. Waiting for mgr to connect...
<date><time> C000056> PPS manager has connected.
<date><time> C000056> Queue reader is started. Pid = 81
<date><time> C000056> .. SST init done.
<date><time> C000056> Startup complete.
```

► **NOTE:** *The log is circular and wraps to the top when it reaches its maximum size of 10 MB.*

2. Navigate through this (or any other) log as follows:

- Press [Enter] or [Spacebar] to scroll through the log until you see the last entry. Press [Enter] again to exit the log.
- Press “G” to go to the bottom of the log, and then press [Enter] to exit the log.
- Press “q” and then [Enter] to stop viewing the log.

For an Ethernet connection, a second screen shows the initiated sockets. For example:

```
<date><time>: Initiated socket 0
<date><time>: Initiated socket 1
<date><time>: Initiated socket 2
<date><time>: Initiated socket 3
<date><time>: Initiated socket 4
<date><time>: Initiated socket 5
<date><time>: Initiated socket 6
<date><time>: Initiated socket 7
<date><time>: Norand SST/OWL protocol interface Version <version>
<date><time> Trying to locate access point...
./nui_eth: Connecting server id 0
End of Ethernet Log. Press <ENTER>
```

<version> is the version of the wireless network protocol interface.

Watching the Log File

Use this option to view new information added to the log.

1. At the Log file menu, press “2” and then [Enter]. A screen similar to the following appears:

```
Press Ctrl-C to quit

nui_log: WNAS log utility. Modlvl=7.00.
nui_log: Realtime log request sent to server 0.
nui_log: Channel created. Waiting for nui_serv.
nui_log: Server has connected.
nui_log: Showing data for terminal (all
        terminals) realtime. Instance 0.
—
```

New messages appear on the screen as they are added.

2. Press Ctrl-C to stop watching the activity.

Clearing the Log File

1. At the Log file menu, press “3” and then [Enter]. The following message appears:

```
This will clear the
logfiles, stop and restart
the RF. Press 'C' to
continue.
```

2. To clear the log file, press “c” or “C” and then [Enter]. This option deletes the log file, stops the RF communication, restarts it, and then creates a new log file. Note that it may take up to 3 minutes for the software to restart.

To return to the Log file menu without deleting the log file, press a key other than “c” or “C,” and then press [Enter].

RS-232 Log Debug Levels

Each entry in the log begins with the date (month and day) followed by the time. These examples show the information following the time. An initial log for debug level 0 is followed by partial logs for levels 1 and 2.

Debug Level=0

A log similar to the following means the 6950 EGS is properly installed and ready to work:

```
nuiserv> WNAS - - Version <6.99>
nuiserv> Copr. 1991-1998 Intermec. All rights reserved.
nuiserv> System ID: LINUX egs8 1.1.73 #8 Fri Jan 12 13:47:02
CST 1997 i486
nuiserv> Initialization in progress.
nuiserv> Server files are in directory ./
```


Debug Level=1

Debug level 1 shows commands and responses from the access points, controllers, and base radios in the wireless network. It begins with the same information shown for debug level 0. Some additional information is provided and shown in the following example:

```
nuiserv> This is server number 0. Type R

nuiserv> .. Opening message queue ...
nuiserv> .. Spawning tty process...
nuiserv> Child 000375: Exec 128, ./nui_comm. Type 2
nuiserv> Server initialization is Complete.
```

Notice that the first entry shows type “R” (for RS-232). Also notice the child process 000375. Further activity with this process is identified in the log by “C000375” preceding the activity description.

In subsequent examples, Modlvl <6.99> shows the module level for the named function.

The following example show initialization of a controller. The controller version takes the form <name> V<#> where name is the name for the software installed on the controller and <#> is its version. The following example contains a controller name and version.

```
C000375> nui_func Modlvl <6.99>: Parent process ID is 374.  
C000375> IPC OK! Server=0, Term=128, ID=2, Qid=512, rflan=1  
C000375> nui_comm: Asynch driver Modlvl <6.99>  
C000055> .. Opening RF tty ports...  
C000055> .. Norand controller initialization...  
C000055> ... Setting interactive mode ...  
C000055> ... Setting response timer ...  
C000055> ... Setting read response timer ...  
C000055> ... Controller Version QXSASYNC V2.23 3250, 127  
terminals  
C000055> ... Sending terminal enable string ...  
C000055> ... Resetting RF Units ...  
C000055> .. Norand controller initialization complete.
```

The following portion of the log for debug level 1 shows activity with a wireless station.

```
C000055> .. rf power up -> 005P,071  
nuiserv> Unit 5 Powered Up <P,071>  
nuiserv> Child 000077: Exec 71, ./nui_tlnt. Type 3  
C000077> nui_func Modlvl <6.99>: Parent process ID is 47.  
C000077> IPC OK! Server=0,Term=5,ID=3,Qid=0,rflan=0,lancon=R  
C000077> nui_tlnt: TELNET-RAW interface. Modlvl<6.99>. Buf=384.  
C000077> nui_tlnt: Resetting unit 5. Reason:(Power Up <000>). 0  
C000077> nui_tlnt: Unit 5 connected to (localhost).
```

Debug Level=2

Debug level 2 provides the most information. Initialization of the 6950 EGS and the controller shows more messages. Debug level 2 shows messages between the 6950 EGS and the wireless stations.

For illustration, two portions of the log at debug level 2 are supplied here. Compare the following samples to the last part of the sample for level 1 to see the additional detail added. Note the original messages that appear.

[illegible]

Ethernet Log

A second log is available for an Ethernet connection to the wireless system. The following examples show the information following the time. A complete log for debug level 0 is followed by partial logs for levels 1 and 2.

Debug Level=0

An example of the information that appears for debug level 0 is on page 8-5.

Debug Level=1

Debug level 1 begins with the same information shown for level 0, but adds more information about radio power up and IP host connectivity. For example:

```
C000166> .. rf power up -> 014P,001,071 (BINARY)
nuiserv> Unit 1 Powered Up <P,001,071>
nuiserv> Child 000197: Exec 7, ./nui_tln.t. Type 3
C000197> nui_func Modlvl <6.99>: Parent process ID is 165.
C000197> IPC OK! Server=0,Term=14,ID=3,QID=128,rflan=1,lancon=E
C000197> nui_tln.t: TELNET-RAW interface. Modlvl <6.99> Buf=3712
C000197> nui_tln.t: Resetting unit 1. Reason: (Power Up). 0
C000197> nui_tln.t: Unit 1 trying (localhost) ...
C000197> nui_tln.t: Unit 1 connected to (localhost).
nuiserv> Radio reboot, messaging multi-application
nuiserv> .. Found multiunit app
nuiserv> Started multiunit 71. Type 3-P,071
```

Debug Level=2

Debug level 2 contains all the data contained in debug level 1, plus all the wireless data packets in and out of the 6950 EGS. The following excerpt shows activity for a single wireless station, “unit 14.”

```
C000251 .. rf power up -> 014P,001, 071 (BINARY)
nuiserv> in unit 14: 014P,1,43449
nuiserv> Unit 14 Powered Up!
nuiserv> in unit 14: 014a
nuiserv> in unit 14: 014K
nuiserv> in unit 14: 014K
nuiserv> in unit 14: 014a
nuiserv> in unit 14: 014a
nuiserv> in unit 14: 014K1
nuiserv> Child 000265: Exec 71, ./nui_tlnt. Type 3
C000265> nui_func Modlvl <6.99>: Parent process ID is 250.
C000265> IPC OK! Server=1,Term=14,ID=3,Qid=0,rflan=1,lancon=E
C000265> nui_tlnt: TELNET-RAW interface. Modlvl <6.99> Buf=3712
C000265> dbg-nui_tlnt< P,1,43449
C000265> nui_tlnt: Resetting unit 14. Reason:(Power Up). 0
C000265> dbg-nui_tlnt> D/^0D^0APower up^0D^0APress <ENTER>.
C000265> dbg-nui_tlnt> D/^0D^0A0x1B[0;0H0x1B[2J0x1B[7m Norand
Corp. ^0D^0A Rf Terminal ^0D^0A Server V6.06 ^0D^0A
(0014) 0x1B[0m^0D^0AEnter telnet host^0D^0Aor press
enter^0D^0Afor (localhost). ^0D^0A>0x070x070x070x07
```

A more complete log would show commands and responses from other wireless stations interspersed within these entries. Note that *nuiserv* spawned child process 000265 for activity with unit 14. Later entries show this process as “C000265.”

Also note the greater than or less than symbols in *dbg-nui_tlnt* entries. The symbols show the direction of the message, where > = outgoing and < = incoming.

Precise syntax for debug messages varies with the type of terminal emulation and Ethernet connection, but the sample shows the detail to expect at debug level 2.

Abnormal Conditions

The examples on the other pages show normal conditions. Abnormal conditions show more information that may help locate the source of the problem. If the logs and debug information are not enough to isolate the problem, call your Intermecc representative or the Customer Response Center. Have documentation of your software maintenance agreement available, or a purchase order for support.

The following example shows the 6950 EGS having difficulty finding a root access point on the network.

```
<date><time>: Initiated socket 0
<date><time>: Initiated socket 1
<date><time>: Initiated socket 2
<date><time>: Initiated socket 3
<date><time>: Initiated socket 4
<date><time>: Initiated socket 5
<date><time>: Initiated socket 6
<date><time>: Initiated socket 7
<date><time> Norand SST/OWL protocol interface Version <version>
<date><time> Trying to locate access points...
<date><time> ./nui_eth: Connecting server id 0.
<date><time> Trying to locate access point...
<date><time> Trying to locate access point...
<date><time> Trying to locate access point...
!
```

The 6950 EGS logs this message about every 15 seconds until a root access point is located on the network.

Time Test

Use the time test to test the throughput on the RF link between the wireless station and the 6950 EGS. Use this option to test speed in the local environment. It sends blocks of data from the 6950 EGS to the wireless station and keeps track of how many seconds it takes.

You must run the test from a wireless station to produce useful results. To do this, use a VT220 wireless station and connect to the 6920 EGS as the host. Log in as “config” with the password “10425rc”.

To run the time test:

1. At the Monitor Menu, press “3” and then [Enter]. The following screen appears:

```
nui_time - V<9.9>
VT220 Timer

How many test
cycles?
```

<9.9> represents the version number of the time test program.

1. Choose how many cycles for the wireless station to attempt and then press [Enter]. The wireless station reports the response time at the end of the test.
2. After you review the results, press [Enter] to exit the test.

The following factors can affect throughput on the RF link:

- ▶ Changes in the electromagnetic spectrum
- ▶ Your distance from the access point or base radio
- ▶ Building construction materials
- ▶ Proximity to sources of electromagnetic “noise” like electric motors or fluorescent lights
- ▶ Radio traffic from outside sources and your own radio network

Different radios in wireless stations produce significantly different responses in the same environment.

Troubleshooting tip:

A good benchmark may be 100 cycles. Note the time to complete 100 cycles when the system is running “normally.” If users report throughput issues, perform the 100-cycle test again to see if the results significantly differ from your benchmark.

Section 9

Updating the 6950 EGS Software



From time to time we release updates to the 6950 EGS software. To update the 6950 EGS, you use an update kit and an external disk drive attached to the 25-pin port marked "Service Only."

The update kit is a chargeable item. You may order disks as part of your software maintenance agreement. Or, you may order a separate kit with disks and an optional external disk drive for an additional cost. Without a software maintenance agreement, you may also order a kit at a higher cost. Contact your representative for details.

Before you update, record your current network and wireless settings.

To display your current network settings:

1. At the System Menu, press "1" (Config).
2. At the Config Menu, press "1" (IP).
3. At the IP Menu, press "1" (View).

You must have the information to restore the configuration after the update. Section 5 has a place to record it.

To display your current wireless settings:

1. At the System Menu, press "1" (Config).
2. At the Config Menu, press "2" (RF Comm).

To update the 6950 EGS software:

1. Turn the 6950 EGS off.
2. Plug the external disk drive into the port marked "Service Only."
3. Insert disk 1 into the disk drive.
4. Power the 6950 EGS on.
5. Follow the screens' prompts or the instructions provided with the upgrade kit.
6. Reconfigure the 6950 EGS.

You can complete the configuration through a telnet session to the 6950 EGS. The Ethernet connection for this purpose must be Ethernet 2.0. For initial configuration, you need the addresses set at the factory:

Default IP address:	198.70.37.28
Subnet mask:	255.255.255.0
Network address:	198.70.37.0

All wireless devices must be enabled to pass the following DIX 2.0 Ethernet frame types:

- ▶ 875a
- ▶ 875b
- ▶ 875c

Network devices must also pass multicast frames. Monitor the 6950 EGS with debug level=0 to check if others have been added.

[illegible]

This section has some general information for communicating to the wireless stations and specific directions for using internet sockets.

The 6950 EGS supports these communication methods:

- ▶ VT220
- ▶ TNVT
- ▶ TN3270
- ▶ TN5250
- ▶ NORAND® Native Sockets

Appellant's Brief EFS filed 08/03/2010 Page 1357 of 1403 6050 Enterprise Gateway Server User's Guide Application No. 09/084741 10-1

NORAND Native is a simple command language for controlling wireless stations from a host application. Native Sockets let a host computer running a Native application communicate to Native wireless stations through the 6950 EGS via a TCP/IP socket connection. Native Sockets support is included in the 6950 EGS version 6.04 and greater. The remainder of this section explains Native Sockets programming.

Native Sockets Overview

Supported Products

The 6950 EGS supports Native Sockets applications for 1100, 1700, 5055, 5900, 6400, and 6550 terminals, using INTERMEC® 2100 UAPs or 6710 Access Points. It supports synthesized UHF, 2.4 GHz OpenAir, and 900 MHz radio technologies.

The 6950 EGS does not support Native Sockets for crystal UHF and 900 MHz SST systems that use RC4000 Series Base Stations or RC3250 Network Controllers.

Communication Basics

You can write Native Sockets programs for any host PC or workstation supporting Ethernet and TCP/IP. Initiate communication between the host and the 6950 EGS by configuring both with a common port number (see “Configuration” later in this section). The 6950 EGS uses this designated TCP port number to communicate with the host.

You must to meet two requirements to set up communication:

1. Configure wireless stations and 6950 EGS with a common host name. This defines wireless stations to 6950 EGS communication sessions.
2. Configure each wireless station for Native emulation and assign it a unique wireless station number.

As each wireless station powers on, the host application opens a unique socket between the 6950 EGS and the host for the wireless station. The 6950 EGS keeps an internal table mapping wireless stations to socket numbers. However, wireless station numbers are *not* available to the application. It must use socket numbers to identify each wireless station session.

Sockets remain open until the host application closes the session (see Reset command), or someone powers the wireless station off then on again, initiating a new session.

Host Programming

Software developers must be familiar with sockets programming to use the Native Sockets capability. We do not provide programming support for internet stream sockets. A good reference on sockets programming is *UNIX Network Programming* by W. Richard Stevens, ISBN 0-13-94876-1.

Coding the Application

See the *Native Terminal Emulation Asynchronous Programmer's Reference Guide* (hereafter called the programmer's guide) listed in Section 1 for details about Native Command Syntax. The 6950 EGS supports wireless station commands in the guide, but does not support the controller commands. The application reads and writes directly to the wireless stations through the 6950 EGS.

The Native protocol was originally written for early model INTERMEC controllers that supported serial host communications. Addition of the sockets communication capabilities to the 6950 EGS required use of a modified Native Syntax. A cross reference is provided in the table that follows. Letter conventions are excerpted from the programmer's guide. All commands are terminated with a carriage return <cr>. The 6950 EGS implementation maintains the 200 character maximum message length limitation documented in the guide.

Table 10-1
Abbreviated Commands

Command	6950 EGS Native Syntax	Original Syntax
Set Control Parameters	eDo/c	StDo/c
Write Terminal Audio	Bd	WtBd
Write Terminal Data *	D/ D/\	WtD/ WtD/\
Reset Terminal	g	Gt
Get Terminal Version	V	DtV
Echo Back	not supported	DtE/

* All "D/" and "D/\ " Command Extensions are supported except Send Multiple Write Displays, "D//".

Response Formats

The programmer's guide lists response formats for each command. However, with Native Sockets the wireless station numbers are not returned in the response string.

For example, a keyboard response to a "Do/x"<cr> command is not in the form "Krd"<cr>. A response with data from a bar code scanner is "Srbcn"<cr>. See the "WtDo/x" command in the programmer's guide for description of the parameters.

Error Messages

The Native command syntax is processed partly by the 6950 EGS and partly by the wireless station. For example the Write Display command has an options list "o" and message content "x" in the form "Do/x"<cr>. The 6950 EGS processes content preceding the forward slash while the wireless station processes content following the slash.

Error responses from the 6950 EGS take the form "?x"<cr>, where "x" is a numeric error code ranging from 1 to 11.

Table 10-2
Write Display Error Codes

Code	Error
1	Syntax error
4	Message plus command exceeds 200 character limit
5	Message plus command sent to disabled wireless station
6	Gap error (timeout between start of message and carriage return)
8	Data check error
10	No more buffers available
11	Sequence error

The wireless station's error responses depend on the specific command. See the programmer's guide for the specific command that produced the error.

Normal Communication

Each wireless station is identified by its number and a separate host session is initiated for communication with it. This works regardless of the operating mode of the wireless station: terminal emulation or Native.

Coding the Application

Code does not need to be designed to handle multiple wireless stations. If two operators select the same application number, a copy of the application runs for each wireless station.

Syntax checking is not done on messages transmitted through the 6950 EGS to the wireless station. Check the response to an asynchronous command for error notifications from the controller.

NORAND Native Syntax for Wireless Stations

See the programmer's guide for details about the syntax.

Write Display

You may use any "WD" command specified in the programmer's guide. We strongly recommend that you familiarize yourself with these commands before writing any code.

For Write Display commands, the communication driver in the 6950 EGS adds the “W.” If the “W” is included here, the message is rejected.

The 6950 EGS allows three additional abbreviated Native commands for: “StD”, “DtV”, and “Gt”.

Other Commands and Responses

The following table shows commands you can send in Native mode and a description of the normal responses from the wireless station.

Table 10-3
Abbreviated Commands

Feature	Description
“e” or “s” = St	Sends the “St” command. Note that you need to send from the “D” on when issuing this. For example, if you were transmitting to wireless station number 2, sending “eD” would send “S002D”. The next response would be “CO”.
“v” = DtV	Sending a “v” requests the version number of the wireless station. That is, sending a “v” issues a DtV command to the wireless station. The next message from that wireless station should be its software version and wireless station number. Internet socket applications return only a version number.
“g” = Gt	Sending a “g” resets the wireless station. The next message from that wireless station should be a power-up message.

Responses received from the wireless stations follow the responses listed in the programmer’s guide. (See the programmer’s guide for more information about the format of incoming messages.) In addition, the 6950 EGS can return codes in the wireless station’s number field. They are explained in the next topic.

Native Mode Responses from the 6950 EGS

The 6950 EGS sends the following codes in the wireless station number field returned to the application.

Table 10-4
Codes in Wireless Station Number Field

Code or Message	Meaning to the Application Program
800	No messages are available for the application. The application can perform housekeeping tasks or go back to waiting for a message (same as "NUI_TIMEOUT" that follows).
998	The wireless station has not passed a message for a predetermined length of time. The application can either ignore the message or act on it. For example, this code could be returned if the wireless station is left unattended with the application running (same as "NUI_DEAD" that follows).
999	The 6950 EGS wants it to terminate. If the application does not terminate within 15 seconds, it is terminated automatically. The application receives this message when the 6950 EGS shuts down or is reset, or if the 6950 EGS detects an error (same as "NUI_STOP" that follows).
NUI_OK	The 6950 EGS completion was normal.
NUI_STOP	The 6950 EGS wants to terminate (same as 999 — see preceding message).
NUI_DEAD	The associated wireless station has not been passed a message for a predetermined length of time (same as 998 — see preceding message).
NUI_TIMEOUT	No data available from the application to be read from the 6950 EGS (same as 800 — see preceding message).
NUI_BAD	A fatal error occurred which prevents continued execution.
NUI_BOOT	The associated wireless station is powered on and can be used to reload data into the wireless station or it can be ignored.

Program interfaces return the wireless station number and the controller or wireless station response. If one of these codes is not set, the field contains the number of the wireless station.

Configuration

You must configure IP, RF communication, and wireless station parameters to support Native Sockets.

To configure IP and RF communication parameters:

1. For IP setup, see page 5-4, “Using the IP Menu.”
2. At the Config Menu, press “2” to configure RF communication. On the Comm Setup menu, do the following:
 - ▶ Set the type to E for an Ethernet connection.
 - ▶ Use the same host name configured in the wireless stations. The host name must be four or fewer letters and is case sensitive.
 - ▶ Set automatic log in to the host.
 - ▶ Set the LAN ID.

The following RF communication setup is properly configured for internet sockets:

```
Comm Setup
1. Type: E (E,R)
2. Lan: 0 (0-7)
3. Host: HOST
4. Rec: 0 (0,1)
5. Auto: 1 (0,1)
6. Dbg: 0 (0,1)
Option: (1-6,E)
```

For help with setting these parameters, see Section 6, “Configuring Radio Frequency Communication.”

3. Exit the Comm Setup menu and then press “3” to configure the RF units (wireless stations).

Modify the IP Host and port information for each wireless station so it corresponds to the host where your NORAND Native application resides and the <portno> in the stream sockets application. “Portno” is the port defined on your host (usually in the services file) for the wireless stations.

The following shows a setup for one wireless station, number 42.

```
nui_cfg--Wire Network Access Server Configuration--Modlvl <version>

RF Unit Setup:

1. Unit number: 42

3. Comment.....Native Terminal

5. IP Host name: DEMO
6. Port.....: 3325
7. User ID.....:
8. Password ....:
9. Copy setup from unit ???
10. Duplicate this setup to unit ??? thru unit ???

CTRL-> Enter option: (E=exit, N=next, P=Previous or item number to change)
HELP->
```

Press:

- 1 To assign a unique number from 0–126 for each wireless station.
- 3 To enter a comment (optional) to identify the type.
- 5 To enter the name of the computer running the Native Sockets program.
- 6 To configure a different port number for Native Sockets. Port 23 is the well-known port for telnet. Supply the new number.
- 7 & 8 The user ID and password are not used with Native Sockets.
- 9 To use a configured wireless station number in option 9 to copy that configuration to this wireless station.
- 10 To copy this setup to other wireless station.

For help, see page 5-25, “Setting Up the RF Units.”

Once everything is configured correctly and the wireless station is powered on, a unique connection (socket) is established between the wireless station and your sockets-based Native emulation program.

Commands to the wireless stations must be in appropriate syntax for their Native mode. A good source for these commands is the programmer's guide available from us. Contact your Intermec representative to order the guide. The instructions for the wireless stations are helpful, but the commands for the controllers and bases should be ignored.

The following Write Display command is a sample startup menu for a wireless station with viewing screen size set to 16.

```
DBCDNL1H2P15/01 - STDIO TEST 02 - DIRECT TEST
```

Notice that this message does not have the “W” at the beginning, nor does it contain the wireless station number. The communication driver in the 6950 EGS adds the “W” and the wireless station number.

Demonstration Program

Following is a “C” demonstration host program for Native sockets. The program uses internet sockets to send Native mode syntax to an INTERMEC wireless station via the 6950 EGS. The program sends key presses made by a wireless station user back to their display.

```

/*****
*****
**
**  nui_sock.c
**
** This is a demo program that uses sockets to send/receive data**
** to/from a Native Mode terminal via the nui_tlnr redirector **
** application.
**
** Usage:  ./nui_sock <portno>
**
*****
*****/

#include <stdio.h>
#include <errno.h>

#ifdef WIN32
#include <windows.h>
#define FD_SETSIZE 256 /* set to number of sockets to open */
#include <winsock.h>
#else
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#endif

#define VERSION "1.00"

unsigned int listen_fd;
unsigned short ipport;

char work_buf[1024];
char tdata[4096];

unsigned char param[]="eD/@ OG00";

int shut_down=0; /* true if program shutdown requested */

struct sockaddr_in cli_addr;
struct hostent host_rec;

fd_set read_fd_set;
fd_set main_fd_set;

```

```

/*
** linked list management stuff
*/
struct LIST {
    unsigned int    fdin;
    unsigned int    fdout;
    char            host[32];
    char            ip[32];
    char            sic[8];
    unsigned short   port;
    unsigned         msgcount;
    struct LIST      *next;
};

struct LIST  *head; /* first pointer in linked list */
struct LIST  *tail; /* last pointer in linked list */
struct LIST  *curr; /* always points to the current socket
                    pointer */

int  main(int argc,char **argv );
int  process_port(void);
int  process_socket(void);
int  add_fd(unsigned int fd,char *host,char *ip);
int  delete_fd(unsigned int fd);

/*****/
main(argc, argv)
int argc;
char  *argv[];
{
    struct sockaddr_in  serv_addr;
    int x=0;
#ifdef WIN32
    WSADATA WsaData;
#endif

```



```
/*
** check the command line arguments
*/
if(argc < 2)
{
    printf("\nnui_sock: Wrong number of arguments.");
    printf("\nnui_sock: Usage: nui_sock <portno>\n\n");
    fflush(stdout);
    exit(1);
}

/*
** get the port number
*/
strcpy(work_buf,argv[1]);
if(strlen(work_buf) < 1)
{
    printf("\nnui_sock: Invalid portno! Must be numeric and >
    0.");
    printf("\nnui_sock: Usage: nui_sock <portno>\n\n");
    fflush(stdout);
    exit(1);
}

ipport=atoi(work_buf);

/*
** set up curses
*/
for(x=0;x<53;x++)
{
    printf("\n");
}
fflush(stdout);
printf("\nnui_sock: Native Socket Demo - Version %s
",VERSION);
fflush(stdout);
```

```
/*
** startup sockets
*/
#ifdef WIN32
    x = WSASStartup (0x0101, &WsaData);
    if (x==SOCKET_ERROR)
    {
        printf("\nnui_sock: can't init winsock interface
        %d.",errno);
        fflush(stdout);
        exit(1);
    }
#endif

/*
* Open a TCP socket (an Internet stream socket).
*/
    FD_ZERO(&main_fd_set);
    if ((listen_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        perror("\nnui_sock: can't open local socket");
        exit(1);
    }

/*
* Fill in the structure "serv_addr" with the address of the
* server that we want to connect with.
*/
    memset(&serv_addr,0x00,sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(ipport);

    if (bind(listen_fd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr)) < 0)
    {
        perror("\nnui_sock: can't bind to local socket");
        exit(1);
    }
    listen(listen_fd, 5);

/*
** add this fd to our work list so we know when clients connect
*/
    FD_SET(listen_fd, &main_fd_set);
```

```
/*
** we are connected, lets see if this thing can work
*/
printf("\nnui_sock: Listen on Port %d\n",ipport);
fflush(stdout);
process_port();

/*
** shut down the system
** (traverse the "connected" list to shutdown each terminal)
*/
curr = head;
while(curr != NULL)
{
    /* this will shutdown and reset the terminal */
    delete_fd(curr->fdin);
    curr = curr->next;
}

shutdown(listen_fd,2);
close(listen_fd);

printf("\nnui_sock: System Shutdown.");
fflush(stdout);

exit(0);
}

/*****/

/*
** main function loop for handling the socket calls from the
** cradle server
*/

int process_port()
{
    int hdnrcnt=0;
    int x=0;
    int clen=0;
    struct timeval seltime;
    int gotdata=0;
    struct hostent *host_rec;
    char host[32];
    char *ip;
    int sockfd;
    int incnt=0;
```

```
/*
** use the select to see if anything is waiting
*/
for(;;)
{
    if (shut_down)
    {
        break; /* user requested shutdown, so goodbye */
    }
    memcpy((void *)&read_fd_set, (void *)&main_fd_set,
        sizeof(fd_set));
    seltime.tv_usec = 0;
    seltime.tv_sec = 15; /* 15 second timeout */

    gotdata=select(FD_SETSIZE,&read_fd_set,0,0,&seltime);

    if(gotdata > 0)
    {
/*
** it is a new connection, connect it
*/
        if(FD_ISSET(listen_fd,&read_fd_set))
        {
            clilen = sizeof(cli_addr);
            sockfd = accept(listen_fd,(struct sockaddr
                *)&cli_addr,&clilen);
            if(sockfd < 0)
            {
                printf("\nnui_sock: Accept error. Port=%d
                    (%d)",ipport,errno);
                fflush(stdout);
            }

            ip=inet_ntoa(cli_addr.sin_addr);
            host_rec=gethostbyaddr((char *)
                &cli_addr.sin_addr,4,AF_INET);
            strcpy(host,host_rec->h_name);

            printf("\nnui_sock: HEARD <%s:%s>",host,ip);
            fflush(stdout);

            FD_CLR(listen_fd,&read_fd_set);

            add_fd(sockfd,host,ip); /* add the connection to
                our fd list */
        }
    }
}
```

```

        /* send terminal control parameters */
        strcpy(tdata,param);
        write(sockfd,tdata,strlen(tdata));

        sleep(1); /* Wait, don't send the next one too
                   fast */

        /* send first entry prompt */

        sprintf(work_buf,"DBSCQDELOH16P16/%-16.16s%-16.16s","Enter data
        now:"," ");
        strcpy(tdata,work_buf);
        write(sockfd,tdata,strlen(tdata));
    }
    else
    {
        /*
        ** look for an active descriptor and process accordingly
        */
        curr = head;
        while(curr != NULL)
        {
            if(FD_ISSET(curr->fdin, &read_fd_set))
            {
                if(!process_socket())
                {
                    /* this will shutdown/reset the
                    terminal */
                    delete_fd(curr->fdin);
                    curr = NULL;
                    break;
                }
            }
            curr = curr->next;
        }
    }
}

return(1);
}

```

```
/*
** This function reads data from the socket (response from the
** terminal) and writes to the socket (sends commands to the
** terminal).
*/
int process_socket()
{
    int datacnt=0;

/*
** read the data from the socket
*/
    memset(tdata,0x00,sizeof(tdata));
    datacnt=read(curr->fdin,tdata,sizeof(tdata));
    if (datacnt < 1)
    {
        printf("\nnui_sock: Bad message. fd=%d,
            datacnt=%d",curr->fdin,datacnt);
        fflush(stdout);
        return(0);
    }

/*
** we have a data message, process it
*/
    if(tdata[0]=='K' && tdata[1]=='Q') /* stop and reset this
        terminal */
    {
        printf("\nnui_sock: Terminal requested stop.");
        fflush(stdout);
        return(0);
    }

    if(tdata[0]=='K' && tdata[1]=='X')
    {
        /* Stop this application and reset all connected terminals
        */
        printf("\nnui_sock: Program stop requested.");
        fflush(stdout);
        shut_down = 1;
        return(0);
    }

    if(tdata[strlen(tdata) - 1] == 13) /* don't forget to strip
        cr */
        tdata[strlen(tdata) - 1] = 0;
```

```
/*
** I'm not doing anything with the incoming data but sending it
** back, thus I build my new output message from the input
** message
*/

    sprintf(work_buf, "DBSCQDEL0H16P16/%-16.16s%-16.16s%-16.16s
        %-16.16s", "Enter data now:", " ", "Previous data:", &tdata[1]);
    strcpy(tdata, work_buf);

/*
** write the data to the socket
*/
    write(curr->fdin, tdata, strlen(tdata));
    return(1);
}

/*
** This function adds a new socket File Descriptor to our
** linked list of connected FDs (which are actually terminals).
*/

int add_fd(unsigned int fd, char *host, char *ip)
{
    struct LIST *ptr;
    ptr = (struct LIST *)malloc(sizeof(struct LIST));
    if(ptr == NULL)
    {
        printf("\nnui_sock: add_fd Cannot allocate buf ptr.");
        fflush(stdout);
        return(-1);
    }

    ptr->fdin = fd;
    strcpy(ptr->ip, ip);
    strcpy(ptr->host, host);
    ptr->next = NULL;
    if(tail != NULL)
    {
        tail->next = ptr;
        tail=ptr;
    }
}
```

```
    else
    {
        head = ptr;
        tail = ptr;
    }

    FD_SET(fd, &main_fd_set);

    return(1);
}

/*
** This function removes a socket File Descriptor from our
** linked list and sends a reset to the terminal to shut
** it down properly through the nui_tlnr program.
*/

int delete_fd(unsigned int fd)
{
    struct LIST *before;
    struct LIST *ptr;
    int foundflag = 0;

    ptr = head;
    before = NULL;

    while(ptr != NULL)
    {
        if(ptr->fdin == fd)
        {
            foundflag = 1;
            break;
        }
        before = ptr;
        ptr = ptr->next;
    }

    if(!foundflag)
    {
        printf("\nnui_sock: delete_fd Tried to delete an fd not
        in list!");
        fflush(stdout);
        return(-1);
    }

    FD_CLR(ptr->fdin, &main_fd_set);
    if(ptr->fdin > 0)
    {

```



```
/*
** sending a 'g' will reset the terminal and close connection
** with nui_tlnr
*/
    tdata[0]='g';    /* reset this radio only */
    tdata[1]=0;
    write(ptr->fdin,tdata,strlen(tdata));
    sleep(1); /* don't shutdown the socket too fast */
    shutdown(ptr->fdin,2); /* shutdown this socket connection
    */
    close(ptr->fdin);
}

if(before != NULL)
{
    before->next = ptr->next;
    if(ptr == tail)
    {
        tail = before;
    }
}
else
{
    head = ptr->next;
    if(head == NULL)
        tail = NULL;
}

free(ptr);
return(1);
}

/*
***** last line of program */
```


INDEX

NOTES

Page numbers in *italics* are figures, those in **bold** are tables.

NUMBERS

- 100BASE-T
 - components, *3-15, 3-15*
 - connecting 6950 EGS to, *3-18, 3-19*
 - connector, *3-9, 3-10*
- 10BASE-T
 - components, *3-15, 3-15*
 - connecting 6950 EGS to, *3-18, 3-19*
 - connector, *3-9, 3-10, 3-11*
- 10BASE2
 - components, *3-14, 3-14, 3-15*
 - connecting 6950 EGS to
 - end, *3-16, 3-17*
 - middle, *3-16, 3-18*
 - connector, *3-9, 3-11*
 - note for cable length, *3-16*
- 10BASE5, connector, *3-9, 3-11*
- 2100 UAP. *See* Access point (2100 and 6710)
- 6710 Access Point. *See* Access point (2100 and 6710)
- 800 message, radio terminal number field, **10-8**
- 998 message, radio terminal number field, **10-8**
- 999 message, radio terminal number field, **10-8**

A

- Abnormal conditions, *8-14*
- AC input port, *3-20, 3-21*
- AC power connection, *3-20, 3-21*

- Access point (2100 and 6710)
 - connecting to network, *3-1, 3-3*
 - connection methods, *2-1*
 - LAN ID, *6-3*
 - manuals, *1-3*
- Adding a host name, *5-20*
- Annunciators, *3-8, 3-8*
- Applying power, *3-20*
- AS/400 host, *3-4*
- AUI connector, *3-11*
- Automatic reconnect menu
 - option, *6-2, 6-7*
- Automatically logging in, *6-4, 6-8*

B

- Back panel of 6950 EGS, *3-9, 3-10, 3-11*
- Baud rate, *4-5, 6-10*
- BBS, *2-6*
- BNC connector, *3-11*
- Booting the 6950 EGS, *7-4*
- Bps, *4-5*
- Bracket, *3-12*
- Bulletin board service, *2-6*

C

- Cable terminator, *3-15, 3-16, 3-17*
- Cable with RJ45 plugs, *3-15*
- Cabling, *3-14*
- Call sign, FCC, *6-10*
- Clearing the log file, *8-7*
- Codes in radio terminal number field, **10-8**
- Coding host application, *10-6*

- Collecting equipment, *3-14*
- COM1 connector
 - cable for, *3-14*
 - connecting to RC3250
 - Network Controller, *3-5, 3-12*
 - connecting to RC4030E
 - Gateway, *3-12*
 - connecting to RCB4030
 - Base/Controller, *3-12*
 - location, *3-10, 3-11*
- COM2 connector
 - connecting dumb terminal to, *3-12, 4-5*
 - location, *3-10, 3-11*
- Comm Setup menu, *6-2, 6-7*
- Commands, abbreviated, **10-4, 10-7**
- Comments, entering, *5-25, 5-27, 10-10*
- Components, Ethernet LAN, *3-14*
- Config Menu, *5-3, 6-2*
- Config RS232 menu option, *6-7*
- Configuring
 - Ethernet connection, *6-2*
 - network options, *5-1*
 - RF communication, *6-1*
 - RS-232 connection, *6-6*
 - via dumb terminal, *4-5*
 - via telnet, *4-2, 9-2*
 - via video monitor and keyboard, *4-4*
- Connecting
 - 6950 EGS to
 - 100BASE-T, *3-18, 3-19*
 - 10BASE-T, *3-18, 3-19*
 - end of 10BASE2, *3-16, 3-17*
 - middle of 10BASE2, *3-16, 3-18*
 - access point to network, *3-3*
 - RC3250 Network Controller to 6950 EGS, *3-4*

Connecting (Continued)

- RB4030 Base Radio to network, 3-3
- RC4030E Gateway to 6950 EGS, 3-4
- Connectors, 6950 EGS, 3-9
- Control Menu, 7-2
- Copy setup, 5-25, 5-27, 10-10
- Customer Response Center, 2-5
- Customer support, 2-4

D

- Data bits, 4-5
- Date and time, 5-28
- Debug
 - description of levels, 6-4, 6-8
 - Ethernet level 0, 8-4, 8-12
 - Ethernet level 1, 8-4, 8-12
 - Ethernet level 2, 8-4, 8-13
 - menu option, 6-2, 6-7
 - RS-232 level 0, 8-4, 8-8
 - RS-232 level 1, 8-4, 8-9
 - RS-232 level 2, 8-4, 8-10
- Default gateway
 - enabling, 5-10
 - in sample network, 5-10, 5-10
 - IP address, 5-2, 5-11, 5-22
- Default host
 - connecting to, 7-8
 - description, 5-9
 - in sample network, 5-10, 5-10, 5-12, 5-12
 - IP address, 5-2, 5-9, 5-25, 5-26, 10-10
 - name, 5-25, 5-26
 - pinging, 5-23
 - reaching, 5-11
 - to which wireless stations attach, 5-26, 10-10
- Default settings
 - automatic login, 6-4, 6-8
 - baud rate, 6-10
 - debug level, 6-4, 6-8
 - DNS host name, 5-7
 - host name, 6-3
 - IP address, 4-2
 - LAN ID, 6-3
 - network IP address, 4-2
 - reconnection, 6-4, 6-8

- subnet mask, 4-2
- type of network, 6-3, 6-8
- wireless network host name, 6-3
- Deleting a host name, 5-21
- Deleting the log file, 8-7
- Demonstration program, Native Sockets, 10-12
- Destination IP address, 5-2, 5-13, 5-22
- DIX 2.0 Ethernet frame types, 5-2, 9-2
- DNS
 - creating a name, 5-6
 - host name, 5-2, 5-7
 - in sample network, 5-12, 5-12
 - IP address, 5-2, 5-15, 5-22
 - secondary IP address, 5-2, 5-16, 5-22
 - viewing settings, 5-22
- Domain name server. *See* DNS
- DtV, **10-7**
- Dumb terminal
 - configuring 6950 EGS through, 4-5
 - keyboard connector, 3-10, 3-11
 - video connector, 3-10, 3-11
- Duplicate setup, 5-25, 5-27, 10-10

E

- End of 10BASE2 segment, 3-17
- Enterprise Wireless LAN, 2-1
- Error codes, write display, 10-5
- Ethernet
 - 100BASE-T connector, 3-10
 - 10BASE2 connector, 3-11
 - 10BASE5 connector, 3-11
 - 10BASE-T connector, 3-10, 3-11
 - 2.0, 5-2, 9-2
 - Comm Setup menu, 6-2
 - configuring RF communication, 6-2
 - connecting to, 3-16
 - examples of log debug levels, 8-12
 - frame types, 5-2, 9-2

- server log, 8-5
- system monitor, 8-3
- External keyboard connector, 3-10, 3-11

F

- Factory service, 2-5
- Fault tolerance, 3-2
- FCC call sign, 6-10
- Figures
 - 100BASE-T connection, 3-19
 - 10BASE-T connection, 3-19
 - 10BASE2 connection, 3-17, 3-18
 - 6950 EGS and access point connected to network, 3-3
 - 6950 EGS and RB4030 Base Radio connected to network, 3-3
 - 6950 EGS connected to RC3250 Controller, 3-5
 - 6950 EGS connected to RC4030E Gateway, **3-7**
 - 6950 EGS connected to RCB4030 Base/Controller, **3-7**
 - 6950 EGS linked by router, 3-4
 - AC power connection, 3-21
 - back panel of current 6950 EGS, 3-10
 - back panel of early 6950 EGS, 3-11
 - cable terminator, 3-15
 - cable with RJ45 plugs, 3-15
 - default gateway to host, 5-10
 - front panel, 3-8
 - RC3250 Controller connected to modem, 3-6
 - static route to host, 5-12
 - T-connector, 3-14

- Frame types network must pass, 5-2, 9-2
- Front panel, 3-8, 3-8

G

- Gateway. *See* Default gateway; RC4030E Gateway

Gateway address, 5-14, 5-22

Gt, **10-7**

H

Halt menu option, 7-2

Halting the 6950 EGS, 7-3

Hard drive light, 3-8

HD light, 3-8

Help, telephone, 2-5, 2-6

Host

See also Default host

AS/400, 3-4

communication basics, 10-2

communications, 2-1

connecting to, 3-1

default gateway to, 5-10

IBM, 3-4, 4-1

login, 7-9

pinging, 5-23

programming, 10-3

static route to, 5-12

UNIX, 3-3, 4-1

VAX, 3-3, 4-1

Host application program, 10-1

HOST Menu, 5-19

Host name

adding to host table, 5-20

configuring for host table,

5-19

deleting from host table, 5-21

DNS, 5-2, 5-7

for Native Sockets, 10-9

viewing host table, 5-22

wireless network, 5-2, 5-7, 6-3

Host table, 5-19, 5-22

I

IBM host, 3-4, 4-1

Indicators, 3-8, 3-8

Inspection, 2-4

Installation, 3-1

Installation equipment, 3-14

Introduction, 2-1

IP address

6950 EGS, 5-2, 5-8

configuring addresses, 5-5,

5-6

default for initial

configuration, 4-2

default gateway, 5-2, 5-11,

5-22

default host, 5-2, 5-9, 5-25,

5-26, 10-10

destination, 5-2, 5-13, 5-22

DNS, 5-2, 5-15, 5-22

local, 5-2, 5-10, 5-12, 5-14,

5-22

network, 4-2, 5-2, 5-8, 5-22

secondary DNS, 5-2, 5-16,

5-22

using to build host table, 5-19

IP configuration, viewing, 5-21

IP Menu, 5-5

IP View menu, 5-21

K

Keyboard and video monitor, 4-4

Keyboard connector, 3-10, 3-11

L

LAN ID

access point, 6-3

configuring, 6-3

menu option, 6-2

RB4030 Base Radio, 5-2, 6-3

wireless station, 6-3

Lights, 3-8, 3-8

Local host table, 5-19

Local IP address, 5-2, 5-10, 5-12,

5-14, 5-22

Localhost, 5-26, 7-8

Location, 3-2, 3-12

Log file menu, 8-4, 8-4

Login, daily, 7-7

Logs

clearing, 8-7

debug levels, 6-4, 6-8, 8-4

deleting, 8-7

examples of debug levels, 8-8,

8-12

viewing, 8-5

watching, 8-7

M

Maintenance, 2-5

Middle of 10BASE2 segment,

3-18

Modem

6950 EGS connected through,

3-5, 3-6

jack on 6950 EGS, 3-10, 3-11

Monitor connector, 3-10, 3-11

Monitor Menu, 8-2

Monitoring the system, 8-1

Mounting brackets, 3-8, 3-12

N

Native mode

commands and responses,

10-7

responses from the server,

10-8

syntax for communicating

with wireless stations,

10-6

syntax source, 10-11

Native Sockets

6950 EGS configuration, 10-9

demonstration program,

10-12

overview, 10-2

supported products, 10-2

user ID, 10-11

Netmask, 5-22

Network

IP address, 4-2, 5-2, 5-8, 5-22

options, 5-1

subnet, 5-8

NUI_BAD message, radio

terminal number field, **10-8**

NUI_BOOT message, radio

terminal number field, **10-8**

NUI_DEAD message, radio

terminal number field, **10-8**

NUI_OK message, radio

terminal number field, **10-8**

NUI_STOP message, radio

terminal number field, **10-8**

NUI_TIMEOUT message, radio terminal number field, **10-8**
 Number field, radio terminal codes, **10-8**

O

Opening a telnet session, 5-24

P

Parity, 4-5
 Password
 6950 EGS login, 4-3
 for configuring wireless stations, 5-25, 5-27
 host login, 7-9
 login, 4-3, 7-9
 Native Sockets, 10-10, 10-11
 Phone jack, 3-10, 3-11
 Phone numbers, 2-5, 2-6
 Ping, 5-5, 5-14, 5-23
 Port
 23, 5-24
 COM1, 3-10, 3-11
 COM2, 3-10, 3-11
 Native Sockets, 10-2, 10-10, 10-11
 TCP, 5-25, 5-26
 Portno, 10-10
 Power
 applying, 3-20
 light, 3-8
 supply socket, 3-10, 3-11
 switch, 3-10, 3-11
 Product support, 2-4
 PS/2 keyboard connector, 3-10, 3-11
 Publications, related, 1-3

Q

Quick start, 2-2

R

Radio frequency communication, 6-1

Radio terminal number field codes, **10-8**
 RB3000 Base Station, 3-1, 3-5, 3-5, 3-6
 RB4030 Base Radio
 connecting to network, 3-1, 3-3
 connection methods, 2-1
 LAN ID, 5-2, 6-3
 RC3250 Network Controller
 connecting to network, 3-5, 3-5, 3-6
 connection methods, 2-2
 RC4030E Gateway, connecting to network, 3-4, 3-7, 3-7
 RCB4030 Base/Controller
 connecting to network, 3-4, 3-6, 3-7
 connection methods, 2-2
 Reboot menu option, 7-2
 Rebooting the 6950 EGS, 7-4
 Reconnect menu option, 6-2, 6-7
 Reconnecting, 6-4, 6-8
 Repair, 2-5
 Required information, 5-1
 Restart menu option, 7-2
 Restarting the 6950 EGS, 7-6
 RF Unit Setup menu, 5-25
 RJ45 jack, 3-10, 3-11, 3-19
 Router
 connecting to host through, 3-4, 3-4
 default gateway to host, 5-10
 destination address, 5-2, 5-13
 gateway address, 5-14
 local address, 5-2
 static route to host, 5-12
 RS-232
 Comm Setup menu, 6-7
 configuring RF communication, 6-6
 connector, 3-10, 3-11
 examples of log debug levels, 8-8
 Setup menu, 6-9, 6-10
 system monitor, 8-3

S

Secondary DNS IP address, 5-2, 5-16, 5-22
 Serial. *See* RS-232
 Server log, 8-5
 Service, factory, 2-5
 Site considerations, 3-2
 Site survey, 3-2
 Specifications, 2-6
 Starting the 6950 EGS, 7-6
 Startup menu, write display command, 10-12
 Static route
 configuring, 5-2, 5-13
 description, 5-12
 to host, 5-12
 viewing settings, 5-22
 STD, **10-7**
 Stop bit, 4-5
 Stopping 6950 EGS, 7-1
 Subnet mask, 4-2
 System monitor, 8-2

T

T-connector, 3-14, 3-14, 3-16
 Telephone jack, 3-10, 3-11
 Telnet
 6950 EGS to host, 5-9
 automatic, 6-4, 6-8
 configuring 6950 EGS through, 4-2, 9-2
 menu option, 5-5
 opening a session, 5-24
 Telnet 3270, 4-1, 5-24, 10-1
 Telnet 5250, 4-1, 5-24, 10-1
 Telnet host, 7-8
 Telnet VT220, 5-24, 10-1
 Terminal, dumb. *See* Dumb terminal
 Thumb screws, 3-8
 Time and date, 5-28
 Time test, 8-15
 TN3270, 4-1, 5-24, 10-1
 TN5250, 4-1, 5-24, 10-1

TNVT, 4-1, 5-24, 10-1

Type option

Ethernet connection, 6-2, 6-3

RS-232 serial connection, 6-7,
6-8

U

Unit number, 5-25, 5-27, 10-10

UNIX host, 3-3, 4-1

Updating the software, 9-1

User ID

Native Sockets, 10-10, 10-11

remote login to IP host, 5-26

V

VAX host, 3-3, 4-1

Video connector, 3-10, 3-11

Video monitor and keyboard, 4-4

Viewing the log file, 8-5

W

Watching the log file, 8-7

WD commands, 10-6

Web site, 2-5

Wireless station

configuring, 5-25, 10-9

host name, 5-7

telnet, 5-24

Write display command, 10-6

Write display error codes, 10-5

MDLC Gateway Communication Server

for Microsoft Windows
and InTouch Applications

**User Manual
Ver 1.x Rev 1.9
DR 21010**

KLINKMANN AUTOMATION
P.O. Box 38
FIN-00371 Helsinki Finland
tel. int. + 358 9 5404940
fax int. + 358 9 5413541
www.klinkmann.com

Table Of Contents

Overview.....	1
Communication Protocols.....	1
Accessing a Remote Items via MDLCGATE Server.....	2
Installing and starting the MDLCGATE Server.....	3
Installing the I/O Server Infrastructure	4
Configuring the MDLCGATE Server.....	6
Server Settings Command.....	6
Gateway Node Definition Command.....	8
Hot Standby - Principles of Operation.....	11
Saving MDLCGATE Configuration File.....	12
Configuration File Location.....	12
Topic Definition Command.....	13
Guidelines on Server Performance	19
Item (Point) Naming.....	23
Historical File.....	26
Using the MDLCGATE Server with InTouch.....	27
Defining the DDE Access names.....	27
Defining the Tag Names	29
Monitoring the Status of Communication with InTouch.....	31
Notes on Using Microsoft Excel.....	32
Reading Values into Excel Spreadsheets.....	32
Writing Values to MDLCGATE Points.....	32
Troubleshooting.....	33
WIN.INI entries	33
Troubleshooting menu	36
MOSCAD System Definition Files	41
RTU Types Definition.....	41
MOSCAD System Definition	43

MDLC Gateway Communication Server

Overview

The **MDLC Gateway Communication Server** (hereafter referred to as the "MDLC Gateway Server" or "MDLCGATE Server" or "MDLCGATE" or "Server") is a Microsoft Windows application program that acts as a communication protocol *Server* and allows other Windows application programs access to data from the MOSCAD Remote Terminal Units, using Motorola MDLC Gateway for TCP/IP. The MDLC Gateway Server requires an Ethernet card and TCP/IP protocol (supporting Windows Sockets interface) installed on the computer to communicate with the MDLC Gateway(s) connected to the Ethernet network.

The MDLC Gateway Server calls MDLC Gateway API routines to establish the connections with Gateway and to send data, commands and data requests to the field RTUs.

The MDLCGATE Server is primarily intended for use with **Wonderware InTouch**, but any Microsoft Windows program that is capable of acting as a **DDE**, **FastDDE** or **SuiteLink Client** may use the MDLCGATE Server.

Communication Protocols

Dynamic Data Exchange (DDE) is a communication protocol developed by Microsoft to allow applications in the Windows environment to send/receive data and instructions to/from each other. It implements a client-server relationship between two concurrently running applications. The server application provides the data and accepts requests from any other application interested in its data. Requesting applications are called clients. Some applications such as Wonderware InTouch and Microsoft Excel can simultaneously be both a client and a server.

FastDDE provides a means of packing many proprietary Wonderware DDE messages into a single Microsoft DDE message. This packing improves efficiency and performance by reducing the total number of DDE transactions required between a client and a server. Although Wonderware's FastDDE has extended the usefulness of DDE for our industry, this extension is being pushed to its performance constraints in distributed environments. The MDLCGATE Server supports the **FastDDE Version 3** -- an extension to Wonderware's proprietary FastDDE Version 2. This extension supports the transfer of Value Time Quality (VTQ) information. The original DDE and FastDDE Version 2 formats are still supported, providing full backward compatibility with older DDE clients. FastDDE Version 3 works on Windows 9x systems as well as Windows NT systems.

NetDDE extends the standard Windows DDE functionality to include communication over local area networks and through serial ports. Network extensions are available to allow DDE links between applications running on different computers connected via networks or modems. For example, NetDDE supports DDE between applications running on IBM

compatible computers connected via LAN or modem and DDE-aware applications running on non-PC based platforms under operating environments such as VMS and UNIX.

SuiteLink uses a TCP/IP based protocol and is designed by Wonderware specifically to meet industrial needs such as data integrity, high-throughput, and easier diagnostics. This protocol standard is only supported on Microsoft Windows NT 4.0 or higher. SuiteLink is not a replacement for DDE, FastDDE, or NetDDE. The protocol used between a client and a server depends on your network connections and configurations. SuiteLink was designed to be the industrial data network distribution standard and provides the following features:

- Value Time Quality (VTQ) places a time stamp and quality indicator on all data values delivered to VTQ-aware clients.
- Extensive diagnostics of the data throughput, server loading, computer resource consumption, and network transport are made accessible through the Microsoft Windows NT operating system Performance Monitor. This feature is critical for the scheme and maintenance of distributed industrial networks.
- Consistent high data volumes can be maintained between applications regardless if the applications are on a single node or distributed over a large node count.
- The network transport protocol is TCP/IP using Microsoft's standard WinSock interface.

The Suite Link, FastDDE (Version 3) and DDE support for MDLCGATE Server is implemented by **Wonderware I/O Server Toolkit** ver. 7.0 (060).

Accessing a Remote Items via MDLCGATE Server

The communication protocol addresses an element of data in a conversation that uses a three-part naming convention that includes the **application name**, **topic name** and **item name**. The following briefly describes each portion of this naming convention:

application name

The name of the Windows program (Server) that will be accessing the data element. In the case of data coming from or going to MOSCAD RTUs, the application portion of the address is **MDLCGATE**.

topic name

Meaningful names are configured in the Server to identify specific devices. These names are then used as the topic name in all conversations to that device. For example, **Node1**. The MDLCGATE Server considers each MOSCAD RTU to be a separate topic

Note. You can define multiple topic names for the same RTU to poll different items at different rates.

item name

Item is a specific data element within the specified topic. For the MDLCGATE Server, an item can be a variable from RTU Communication Table or some special purpose item.

(The item/point names are fixed by the MDLCGATE Server as described in the **Item (Point) Naming** section.)

Note: In some cases, the term "point" is used interchangeably with the term "item".

Installing and starting the MDLCGATE Server

The MDLCGATE Server requires an Ethernet card and TCP/IP protocol installed on the computer to communicate with the Motorola MDLC Gateway. The MDLCGATE Server is tested with 3Com's EtherLink III card.

The MDLCGATE Server can be installed before or after installing the Ethernet card and TCP/IP protocol. Also the HASP key must be plugged into the computer parallel port.

The MDLCGATE Server installation package can be supplied:

1. As a self-extracting archive 21010xxx.EXE if downloaded from Klinkmann's web site (the xxx is the current (latest) version of the Server).
2. From installation on CD.
3. On two or three distribution disks (floppies).

To **install** the MDLCGATE Server from the self-extracting archive, run the 21010xxx.EXE and proceed as directed by the MDLCGATE Server Setup program.

To **install** the MDLCGATE Server from CD or distribution disks, on MS Windows (NT, 2000, XP or 9x):

1. Insert the CD with Klinkmann Software into CD drive or insert the MDLCGATE Server Disk1 into a floppy drive A: or B:.
2. Select the **Run** command under the **Start** menu.
3. Run STARTUP.EXE if installing from CD or SETUP.EXE if installing from distribution disks (floppies).
4. If installing from CD: select "Protocol Servers (DDE, SuiteLink, OPC)", find "MDLCGATE SL and DDE Server" and click on "Setup...".
5. Proceed as directed by the MDLCGATE Server Setup program.

When installation is finished, the subdirectory specified as a folder where to install the MDLCGATE Server files will contain the following files:

MDLCGATE.EXE	The MDLCGATE Server Program. This is a Microsoft Windows 32-bit application program.
MDLCGATE.HLP	The MDLCGATE Server Help file.
MDLCGATE.CFG	An example configuration file.
RTUTYPES.CFG	An example configuration file.
MSYSDEF.CFG	An example configuration file.
LICENSE.TXT	Klinkmann Automation software license file.

To **uninstall** the MDLCGATE Server, start Control Panel, select "Add/Remove Programs" and select the "MDLCGATE SL and DDE Server" from the list of available software products. Click on "Add/Remove..." and proceed as directed by the UninstallShield program.

Notes:

1. The MDLCGATE Server is developed with Wonderware I/O Server Toolkit (ver 7.0) and needs the **Wonderware FS2000 Common Components** to be installed on computer where the MDLCGATE Server is running. The Wonderware FS2000 Common Components are installed automatically when any of Wonderware FS2000 Components (e.g. InTouch or some Wonderware I/O server) is installed.
2. If MDLCGATE Server will run on PC where Wonderware FS2000 Common Components are not installed then a special **I/O Server Infrastructure installation package** can be obtained from Klinkmann Automation (see **Installing the I/O Server Infrastructure** section below). This I/O Server Infrastructure installation package contains the minimum set of software needed to run the MDLCGATE Server and these infrastructure files must be install prior to executing the MDLCGATE Server.
3. The HASP key is needed for full time running of MDLCGATE Server. The HASP Driver setup is performed during the Server setup. Without HASP Driver installed the MDLCGATE Server will run only 1 hour (with all features enabled).

The Server can be started in some different ways:

- (1) At MS Windows startup from Startup group;
- (2) Manually - before client (Wonderware InTouch, MS Excel) startup;
- (3) When starting client (e.g. Wonderware InTouch) with Server name in the path.

At Server startup the MDLC Gateway API function `gwlfb_init_api()` is called to build the internal data structure used to identify the MOSCAD system. The names of two files (RTU Types Definition file and MOSCAD System Definition file) can be specified in the Server command line or default file names RTUTYPES.CFG and MSYSDEF.CFG can be used.

For example, the following command will start the MDLCGATE Server with non-default definition files:

mdlcgate mdlc_typ.cfg mdlc_sys.cfg

where **mdlc_typ.cfg** is RTU Types Definition file and **mdlc_sys.cfg** is MOSCAD System Definition file. The structure of these files is described in the **MOSCAD System Definition Files** section.

Installing the I/O Server Infrastructure

The I/O Server Infrastructure installation package can be supplied:

1. As a self-extracting archive (IOServerInfrastructure.exe) if downloaded from Klinkmann's web site.
2. On one distribution disk (floppy).

To **install** the I/O Server Infrastructure from the self-extracting archive, run the IOServerInfrastructure.exe and proceed as directed by the I/O Server Infrastructure Setup program.

To **install** the I/O Server Infrastructure from the distribution disk, on MS Windows (NT or 9x):

1. Insert the I/O Server Infrastructure disk into a floppy drive A: or B:.
2. Select the **Run** command under the **Start** menu.
3. Type "A:SETUP" or "B:SETUP".
4. Click on **OK**.

5. Proceed as directed by the I/O Server Infrastructure Setup program.

To **uninstall** the I/O Server Infrastructure, start Control Panel, select "Add/Remove Programs" and select the "I/O Server Infrastructure" from the list of available software products. Click on "Add/Remove..." and proceed as directed by the UninstallShield program.

Note. *The I/O Server Infrastructure installation will be rejected if Wonderware FS2000 Common Components are already installed on same computer.*

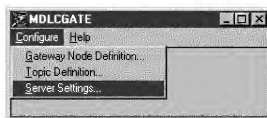
Configuring the MDLCGATE Server

After the MDLCGATE Server is initially installed, a small amount of configuration is required. Configuring the Server automatically creates a **MDLCGATE.CFG** file that holds all of the Gateway Node and Topic definitions entered. This file will automatically be placed in the same directory in which the MDLCGATE Server is located unless the path where the configuration file will be placed is specified via the */Configure/Server Settings...* command.

To perform the required configurations, start up the MDLCGATE Server. If the Server starts up as an icon, double-click on the icon to open the Server's window. The following will appear:



To access the commands used for the various configurations, open the */Configure* menu:

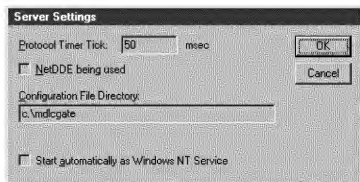


Note: The */Configure/Topic Definition...* command invokes the configuration dialogs for System's *all* Topics (for System's *all* Nodes if multiple gateways are used). To configure Topics on a per Node basis the */Configure/Gateway Node Definition...* command should be invoked.

Server Settings Command

A number of parameters that control the internal operation of the Server can be set. In most cases, the default settings for these parameters provide good performance and do not require changing. However, they can be changed to fine-tune the Server for a specific environment.

To change the Server's internal parameters, invoke the */Configure/Server Settings...* command. The "Server Settings" dialog box will appear:



The following describes each field in this dialog box:

Protocol Timer Tick

This field is used to change the frequency at which the Server is continuously activated (the Server checks for work to do). If the computer is very busy or some other MS Windows application is taking over the computer then the Server could be activated less frequently than the setting in the **Protocol Timer Tick**.

Note: The default value is 50. If a value lower than 50 is entered, the Server uses 50 milliseconds. For Windows NT the minimum value is 10 milliseconds.

NetDDE being used

Select this option if you are networking using NetDDE.

Configuration File Directory

Specify the path (disk drive and directory) in which MDLCGATE Server will save its current configuration file. MDLCGATE Server will use this path to load the configuration file the next time it is started.

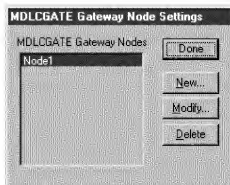
Note: Only the "path" may be modified with this field. The configuration file is always named **MDLCGATE.CFG**.

Note: There is no limit to the number of configuration files created, although each must be in a separate directory. When using the MDLCGATE Server with **InTouch**, it is good practice to place the configuration file in the application directory.

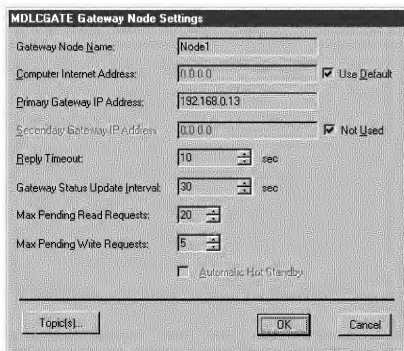
Once all entries have been made, click on **OK**.

Gateway Node Definition Command

To configure the MDLC Gateway Node, invoke the */Configure/Gateway Node Definition...* command. The "MDLCGATE Gateway Node Settings" first dialog box will appear:



To modify or examine an existing Gateway Node, select the Node name and click on **Modify**. To define a new Gateway Node, click on **New**. The "MDLCGATE Gateway Node Settings" second dialog box will appear. The following is an example of "MDLCGATE Gateway Node Settings" dialog box where the default configuration is entered (single Primary Gateway, no Hot Standby):



The following describes each dialog field in this dialog box:

Gateway Node Name

Enter the Gateway Node name and later use it in *Topic Definition*.

Computer Internet Address

Enter the Computer Internet Address (IP Address) if it has more than one. If there is only one Internet Address for computer then **Use Default** can be checked to use this Address. If Computer is multi-homed (more than one Internet Address used) and **Use Default** is checked then it is impossible to know which Address must be used.

Primary Gateway IP Address

Enter the Primary Gateway Internet Address (IP Address).

Secondary Gateway IP Address

Enter the Secondary Gateway Internet Address (IP Address). If **Not Used** is checked then Secondary Gateway is not used for this Gateway Node.

Reply Timeout

Enter the reply timeout value (in seconds) for the TCP/IP communications; this value is used as a *timeout* parameter for the MDLC Gateway API function *gwlib_receive_buffer()*.

Gateway Status Update Interval

Enter the interval (in seconds) at which the Gateway status (both for the Primary and Secondary Gateways) will be read.

Max Pending Read Requests

The possibility to configure the maximum number of pending read requests which are allowed to be sent to the Gateway. This parameter value depends on MOTOROLA Gateway hardware model and current settings. In two Gateways using mode this number cannot exceed weaker Gateway capacity. MDLCGATE Server allows set up to 128 pending read(poll) requests. Smallest possible value is 2(1 Gateway STATUS read(poll) request and one regular read(poll) request). Default value is 20.

Max Pending Write Requests

The possibility to configure the maximum number of pending write requests which are allowed to be sent to the Gateway. This parameter value depends on MOTOROLA Gateway hardware model and current settings. In two Gateways using mode this number cannot exceed weaker Gateway capacity. MDLCGATE Server allows set up to 128 pending write(command) requests. Smallest possible value is 1. Default value is 5.

Automatic Hot Standby

This setting enables or disables the "*automatic hot standby*" to indicate how the hot standby is implemented; if this setting is ON (checkbox is checked) then hot standby is processed by the Server; if this setting is OFF (checkbox is unchecked) then hot standby must be processed by the client program using special items/points created for topic **GATEWAYS** (see the **Hot Standby - Principles of Operation** and **Item (Point) Naming** sections).

Topic(s)...

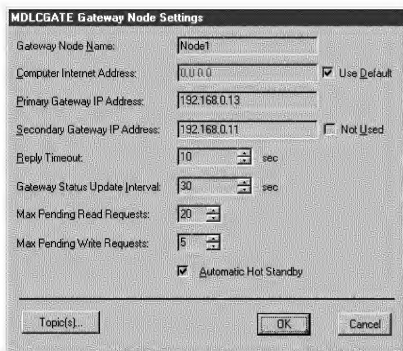
To configure only the current Gateway Node's Topics press the **Topic(s)...** pushbutton. The "Topic Definition" dialog box (see the **Topic Definition Section**) will appear. Up to 500 Topics can be configured for one Gateway Node.

Notes:

1. The channel type opened both for Primary Gateway and (if configured) for Secondary Gateway and used to send/receive poll messages and to receive burst messages is `GWLIB_BURST_CHANNEL_TYPE | GWLIB_SEND_CHANNEL_TYPE | GWLIB_RECEIVE_CHANNEL_TYPE`.

2. The Gateway status is requested at *Gateway status update interval* both for Primary Gateway and (if configured) for Secondary Gateway. The data from all connected RTUs are accessed through the Primary Gateway.

The following is an example of "MDLCGATE Gateway Node Settings" dialog box where the **Secondary Gateway IP Address** is entered and **Automatic Hot Standby** is used:



The image shows a screenshot of the "MDLCGATE Gateway Node Settings" dialog box. The title bar reads "MDLCGATE Gateway Node Settings". The dialog contains several fields and checkboxes:

- Gateway Node Name:** A text field containing "Node1".
- Computer Internet Address:** A text field containing "0.0.0.0" with a checkbox labeled "Use Default" checked.
- Primary Gateway IP Address:** A text field containing "192.168.0.13".
- Secondary Gateway IP Address:** A text field containing "192.168.0.11" with a checkbox labeled "Not Used" unchecked.
- Reply Timeout:** A spin box set to "10" with the unit "sec".
- Gateway Status Update Interval:** A spin box set to "30" with the unit "sec".
- Max Pending Read Requests:** A spin box set to "20".
- Max Pending Write Requests:** A spin box set to "5".
- Automatic Hot Standby:** A checkbox that is checked.

At the bottom of the dialog are three buttons: "Topic(s)...", "OK", and "Cancel".

Once all entries have been made, click on **OK** to process the configuration for the Gateway Node. The "MDLCGATE Gateway Node Settings" first dialog box will appear again.

Click on **Done** when configuration for all Gateway Nodes has been performed.

Note: If this is the first time the Gateway Nodes have been configured, the user will be prompted to save configuration to an existing directory.

Hot Standby - Principles of Operation

The MDLC Gateway redundancy is supported as an option to increase the reliability of the system. Two MDLC Gateways can be configured in a Primary/Secondary mode. If Primary Gateway is not available then Secondary Gateway can be used. Each Gateway has to be configured with unique IP address. Also redundant Hosts (computers with MDLC Gateway Server running) and redundant MDLC and LAN communication devices can be used.

The following are examples of possible configurations:

1. A single Primary Gateway with a single Host. No MDLC Gateway redundancy. No MDLC Gateway Server redundancy - the Server is running only on one computer.
2. A single Primary Gateway with multiple Hosts. No MDLC Gateway redundancy. The redundant MDLC Gateway Servers are used - the Server is running on multiple computers and if some of them fail then data from RTUs can still be accessed by other ones.
3. Dual redundant Primary/Secondary Gateway with a single Host. The MDLC Gateway redundancy supported, no MDLC Gateway Server redundancy
4. Dual redundant Primary/Secondary Gateway with multiple Hosts. The MDLC Gateway redundancy supported, redundant MDLC Gateway Servers are used.

Depending on necessary reliability level the above mentioned dual Gateway configurations can be used with single communication device, with redundant communication devices and with multiple redundant communication devices.

The **Hot Standby** feature, implemented in the MDLC Gateway Server, supports the redundant MDLC Gateways in the following way.

1. After startup the Gateway status is requested at *Gateway status update interval* both for Primary and Secondary Gateways. The data from all connected RTUs are accessed through the Primary Gateway.
2. If the communication with Primary Gateway fails (no answer to *gwlib_get_gw_status()* command or returned Gateway status is Secondary) then depending on the "*automatic hot standby*" setting the hot standby is processed in the following way.

If "*automatic hot standby*" is ON and if communication with Secondary Gateway is O.K. (returned status is *GWLIB_SECONDARY*) then Secondary Gateway is changed to Primary (by MDLC Gateway API function) and data from all connected RTUs now are accessed through this Gateway's channels.

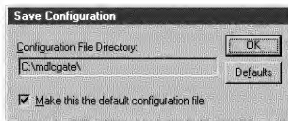
If "*automatic hot standby*" is OFF then the status of each Gateway can be monitored and changed if special topic **GATEWAYS** containing items **GS_LSB_XXX.XXX.XXX.XXX**, **GS_MSB_XXX.XXX.XXX.XXX** and **GMODE_SET_XXX.XXX.XXX.XXX** (where **XXX.XXX.XXX.XXX** is the Gateway IP address) is created (see the **Item (Point) Naming** section). In this case if Primary Gateway fails and if Secondary Gateway status is O.K. then **GMODE_SET_XXX.XXX.XXX.XXX** (where **XXX.XXX.XXX.XXX** is the Secondary

Gateway IP address) value must be switched from 0 to 1 and then Secondary Gateway is changed to Primary.

3. The Primary Gateway can be set to Secondary by switching the Discrete item GMODE_SET_ **xxx.xxx.xxx.xxx** value (where **xxx.xxx.xxx.xxx** is the Primary Gateway IP address) to 0. The Secondary Gateway can be set to Primary by switching the Discrete item GMODE_SET_ **xxx.xxx.xxx.xxx** value (where **xxx.xxx.xxx.xxx** is the Secondary Gateway IP address) to 1.

Saving MDLCGATE Configuration File

If the configuration file does not currently exist, or a new configuration path has been specified, the Server will display the "Save Configuration" dialog box:



This dialog box displays the path where the MDLCGATE Server is going to save the current configuration file. The path may be changed if necessary. Also, the path can optionally be recorded in the **WIN.INI** file by selecting the "**Make this the default configuration file**" option. Doing so will allow the MDLCGATE Server to find the configuration file automatically each time it is started. This option is available only when the configuration file path is changed.

Configuration File Location

When the MDLCGATE Server starts up, it first attempts to locate its configuration file by, first checking the **WIN.INI** file for a path that was previously specified. If the path is not present in the **WIN.INI** file, the Server will assume that the current working directory is to be used.

To start the Server from an application directory configuration file other than the default configuration file a special switch (**/d:**) is used. For example, invoke the **File/Run** command and enter the following:

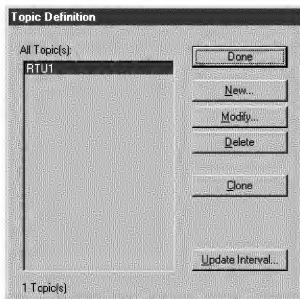
MDLCGATE /d:c:\directoryname

Note: *There is no limit to the number of configuration files that may be created, although each must be in a separate directory.*

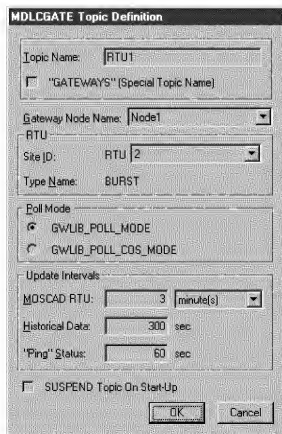
Topic Definition Command

The user provides each MOSCAD Remote Terminal Unit with an arbitrary name that is used as the topic for all references to the RTU. More than one Topic can be defined for the same RTU.

To define the Topics (MOSCAD RTUs) attached to one or more MDLC Gateways invoke the **/Configure/Topic Definition...** command. The "Topic Definition" dialog box will appear:



To modify or examine an existing topic, select the topic name and click on **Modify**. To define a new topic, click on **New**. The "MDLCGATE Topic Definition" dialog box will appear:



The image shows a dialog box titled "MDLCGATE Topic Definition". It contains several fields and options for configuring a topic. The fields are: Topic Name (set to RTU1), Gateway Node Name (set to Node1), RTU (set to RTU), Site ID (set to RTU 2), and Type Name (set to BURST). There are two radio buttons for Poll Mode: GWLIB_POLL_MODE (selected) and GWLIB_POLL_COS_MODE. Under Update Intervals, there are three fields: MOSCAD RTU (set to 3 minute(s)), Historical Data (set to 300 sec), and 'Ping' Status (set to 60 sec). At the bottom, there is a checkbox for SUSPEND Topic On Start-Up (unchecked) and two buttons: OK and Cancel.

The following describes each dialog field in this dialog box:

Topic Name

Enter the Topic Name that corresponds to the Topic Name in the "Access Name Definition" dialog box described in the **Using the MDLCGATE Server with InTouch** section.

"GATEWAYS" (Special Topic Name)

If this checkbox is checked then this is a special topic **GATEWAYS** containing only some special predefined items (see the **Item (Point) Naming** section).

Gateway Node Name

Select the Gateway Node to associate it with the topic. Additional topics may be associated with the same Gateway Node at a later time.

RTU Site ID

Enter this RTU Site ID. The corresponding value must be present in the MOSCAD System Definition file.

RTU Type Name

This RTU Type Name (value from the RTU Types Definition file) is displayed here. The new value can not be entered. This field is used only for information.

Poll mode

Select the *poll mode* (**GWLIB_POLL_MODE** or **GWLIB_POLL_COS_MODE**) used to send poll requests to this RTU.

MOSCAD RTU Update Interval

Enter the frequency (in configurable units (milliseconds, seconds, minutes, hours)) at which the values of items will be read (polled) from this MOSCAD RTU.

Historical Data Update Interval

Enter the frequency (in seconds) at which the values of specially marked items (item/point name with suffix h(or H)) will be stored in the historical file (see the sections **Item (Point) Naming** and **Historical File**). The value is stored only if it has been changed.

"Ping" Status Update Interval

Enter the frequency (in seconds) at which this MOSCAD RTU status is requested. The RTU status "ping" command requests this RTU first Table's (first Table is from the RTU Types Definition file, not Table 1) first element (column 0, row 0). "Ping" request is disabled if value of "Ping" Status Update Interval is 0. Once all entries have been made, click on **OK** to process the configuration for the Topic. The "Topic Definition" dialog box will appear again.

SUSPEND Topic On Start-Up

The MDLC Gateway Server supports the possibility to protect communication from data flow impact at connection start-up. If SUSPEND checkbox is selected then no poll messages of the topic will be sent. State of this "SUSPEND Topic On Start-Up" checkbox is assigned to special "SUSPEND"(see **Item(Point) Naming** chapter) item of the Topic if such item is activated in user's InTouch application. Further communication control can be performed using SUSPEND item of corresponding topic: if value of item is set to 0 then polling is stopped otherwise polling is executed in general way.

Note: This option does not take effect to write messages execution.

Clone of Topic

Since MDLCGATE supports up to 500 topics (per Gateway) the configuration of topics could be time-consuming. It is possible to make clones (similar copies) of any topic (except special GATEWAYS topic). Only difference between clone(s) and base topic is name. Name can be generated by MDLCGATE Server or extracted from text file. To clone some topic select it and press "Clone" button. "Topic Clone" dialog box will appear:

TOPIC CLONE

Topic to Clone: RTU1

Number of Clones: 3

Base Name: SAMPLE

☒ Pad with 0

File Name: Browse...

OK Cancel

Topic to Clone

Appears topic name of cloned topic.

Number Of Clones

Enter the number of copies(clones) will be created.

Base Name

Enter the "Base Name" for created Topic Names. Name of created topic will be: [BaseName]+[NumberOfClone].

Examples:

1) if "Base Name" is SAMPLE and "Number of Clones" is 3 and "Pad with 0" checkbox **is not checked** then will be created 3 topics with following names: SAMPLE1, SAMPLE2, SAMPLE3.

2) if "Base Name" is SAMPLE and "Number of Clones" is 3 and "Pad with 0" checkbox **is checked** then will be created 3 topics with following names: SAMPLE001, SAMPLE002, SAMPLE003.

File Name

To extracted clone name(s) from the text file check the radio button nearby "File Name" static control. "Browse..." button and "File Name" static control and editbox becomes enabled. Do not type the file name into "File Name" editbox but press "Browse..." button to select the file with topic names.

Browse... and file format

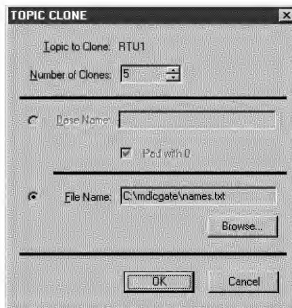
To enter file name press "Browse..." button. "Load TXT file" common dialog box will appear:



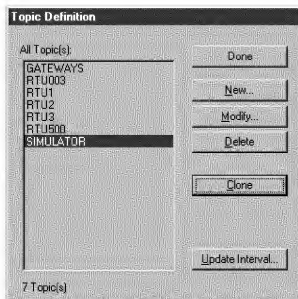
Select the file with topic names. The file with topic names must be plain text file previously created by any text editor (e.g. NOTEPAD). Each line of such file must contain either one topic name or comment. Comment starts by ';' or '#'. Default file extension is *.TXT. The following is contents of example file (names.txt):

RTU003
RTU2
RTU3

RTU500 SIMULATOR

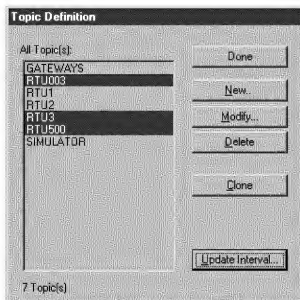


For topic creation press OK. Server will report about successfully created topics (number of clones should be equal to *Number of Clones* configuration value). New topics will appear in the Topic Definition window:

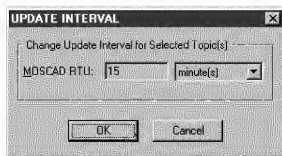


Update Interval for Multiple Topics

The MDLC Gateway Server supports the possibility to enter the new **MOSCAD RTU Update Interval** value for multiple Topics in a single operation. At first these multiple Topics must be selected in the "MDLCGATE Topic Definition" dialog box by holding the CTRL key while clicking on the selected ones:



Then click on **Update Interval**. The "UPDATE INTERVAL" dialog box will appear:



Enter the new value in the **MOSCAD RTU Update Interval** field and click on **OK** to change this value for all selected Topics or click on **Cancel**. The "Topic Definition" dialog box will appear again.

The **MOSCAD RTU Update Interval** can be change also at run-time, i.e. when Topics are activated by client application (see **Guidelines on Server Performance** section).

Select **Done** when configuration for all MOSCAD RTUs has been performed.

Guidelines on Server Performance

Depending on the complexity of the MOSCAD environment where the MDLC Gateway Server is used, the following guidelines should help to achieve the optimum Server performance.

Update Intervals

The optimum usage of Update Intervals is very important. There are three Update Intervals (listed according to their priority) which greatly affect the performance of the Server:

- (1) Gateway Status Update Interval,
- (2) "Ping" Status Update Interval,
- (3) MOSCAD RTU Update Interval.

If Gateway Status and "Ping" Status Update Intervals are very short (less than one second or for very complex systems less than ten seconds) and also data from RTUs are requested very often (MOSCAD RTU Update Intervals also are some seconds), then it is very possible that there is not enough time to execute all poll requests. 20 - 60 seconds could be a reasonable value for Gateway Status and "Ping" Status Update Interval in not too complex systems. Gateway Status poll and RTU Ping are intended for error detection. Theoretically one poll should be equal up to three RTU Ping and up to ten Gateway Status Intervals because it gives some time to Server for error processing. It should protect the server from useless polling(reading) and writing.

The optimum MOSCAD RTU Update Interval value depends on the total number of connected (active) RTUs to which poll requests are sent and also on total amount of active items for the corresponding Topic. It is very difficult to suggest some default value for MOSCAD RTU Update Interval because it highly depends on current environment. This Update Interval must give enough time to execute all poll requests, receive burst data and not to overload the system or the radio airtime usage. Typically RTU Update Intervals in radio-based systems are 10-30 minutes or 1-12 hours depending on system requirements.

Note: *Unlike other SCADA systems MOSCAD uses an unsolicited, spontaneous burst mechanism to update important data changes. Therefore it is less critical to poll RTUs frequently.*

QUEUE_IS_FULL_XXX.XXX.XXX.XXX item

The Server supports Discrete Read Only item **QUEUE_IS_FULL_XXX.XXX.XXX.XXX** (where **XXX.XXX.XXX.XXX** is the Gateway IP address), used for indication of polling queue or write queue overflow. This item is connected to whole active Gateway, not to some RTU (special Topic "GATEWAYS" is used). This item value is OFF (0) if the current number of poll requests for current active Gateway does not exceed **Max Pending Poll Requests** (see chapter **MDLCGATE Gateway Node Settings**) or if the current number of pending write commands does not exceed **Max Pending Write Requests** (see chapter **MDLCGATE Gateway Node Settings**). This item value changes to ON (1) if there are more than maximum pending poll requests or maximum write requests allowed for this Gateway(Node).

If **QUEUE_IS_FULL_XXX.XXX.XXX.XXX** item is specified in the client application (e.g. in InTouch) then the system load can be monitored. If **QUEUE_IS_FULL_XXX.XXX.XXX.XXX**

is ON too long then operator can modify (increase) the MOSCAD RTU Update Interval value for single or multiple Topics (see **Changing MOSCAD RTU Update Interval at run-time** subsection).

RESET_QUEUE_XXX.XXX.XXX.XXX item

For special topic "GATEWAYS" one additional item "RESET_QUEUE_XXX.XXX.XXX.XXX", (where XXX.XXX.XXX.XXX is the IP address of the Primary Gateway of the node) can be used. This is a Read/Write discrete item. If this item value is changed from 0 to 1 then both read and write queue will be reseted. After setting "RESET_QUEUE_XXX.XXX.XXX.XXX" to 1 replies on previous requests are not expected. The capacity of write requests queue becomes equal to "Max Pending Write Requests" value and capacity of read(poll) requests queue becomes equal to "Max Pending Read Requests" value. Value can be switched to 1 and both queues will be emptied. This possibility is usable in case when data exchange is wavelike. Unreachable Gateway or data flow impact can be a reason for non-answering requests. Such situation can be recognized at communication silence when the special topic "GATEWAYS" items READ_COUNTER_XXX.XXX.XXX.XXX does not become equal to "Max Pending Read Request" or WRITE_COUNTER_XXX.XXX.XXX.XXX and does not become equal to "Max Pending Write Request" (see chapter **MDLCGATE Gateway Node Settings** and chapter **Item(Point) Naming**). It means that servers performance is reduced (less and less pending request can be sent) after each non-answered request because Server waits FOREVER answer on the sent request(except Gateway Status message).

RESET_READS_XXX.XXX.XXX.XXX item

For special topic "GATEWAYS" one additional item "RESET_READS_XXX.XXX.XXX.XXX", (where XXX.XXX.XXX.XXX is the IP address of the Primary Gateway of the node) can be used. This is a Read/Write discrete item. If this item value is changed from 0 to 1 then only read part of queue will be reseted. After setting "RESET_READS_XXX.XXX.XXX.XXX" to 1 replies on previous poll requests are not expected and READ_COUNTER_XXX.XXX.XXX.XXX becomes equal to current Gateway Node **Max Pending Read Requests** parameter value (see chapter **MDLCGATE Gateway Node Settings**).

RESET_WRITES_XXX.XXX.XXX.XXX item

For special topic "GATEWAYS" one additional item "RESET_WRITES_XXX.XXX.XXX.XXX", (where XXX.XXX.XXX.XXX is the IP address of the Primary Gateway of the node) can be used. This is a Read/Write discrete item. If this item value is changed from 0 to 1 then only write part of queue will be reseted. After setting "RESET_WRITES_XXX.XXX.XXX.XXX" to 1 replies on previous write requests are not expected and WRITE_COUNTER_XXX.XXX.XXX.XXX becomes equal to current Gateway Node **Max Pending Write Requests** value (see chapter **MDLCGATE Gateway Node Settings**).

SUSPEND item

Discrete Read-Write item. Very important and usable possibility against wavelike data flow impact. This item gives control on data exchanging. The initial value of the SUSPEND item the MDLCGATE Server takes from the "SUSPEND Topic On Start-Up" checkbox (see chapter **MDLCGATE Topic Definition**). If value of item is 0 reads(polls) executes in general way. If SUSPEND item value becomes 1 only writes of this topic will be executed. Read(poll) requests can be executed only by item POLL_NOW using.

POLL_NOW item

The Server supports Discrete Write Only item **POLL_NOW**, used for immediately one time polling of all active items from some RTU. This one time polling can be used when MOSCAD RTU Update Interval value has been changed at run-time and it is necessary to receive fresh data from RTU before previous MOSCAD RTU Update Interval is elapsed. At start-up the value of the **POLL_NOW** item is 0. The Server issued all poll messages of corresponding topic when value of item changes from 0 to 1. The changing of **POLL_NOW** value from 1 to 0 does not take effect.

Note: The Server does not change (resets) value from 1 to 0. This reset operation has to be performed by client (e.g. InTouch).

Changing MOSCAD RTU Update Interval at run-time

At run-time the MOSCAD RTU Update Interval can be changed in two different ways:

- (1) Manually for single or multiple Topics as described in the **Topic Definition Command** section. In this case configurable units (milliseconds, seconds, minutes, hours) can be used.
- (2) In the client application for single Topic by defining the Integer Read/Write item **UPDATE_INTERVAL**. If this item value is changed in the client application (e.g. manually or by InTouch script) then the Server uses this value in the further processing of poll requests. In this case only seconds can be used.

It is important to understand that new MOSCAD RTU Update Interval value becomes effective only after the previous MOSCAD RTU Update Interval has elapsed, i.e. there is no automatic all RTU data polling performed at Update Interval change. This is done in order to prevent additional overload if the system is still overloaded (**QUEUE_IS_FULL_XXX.XXX.XXX.XXX** is ON). To perform immediate poll when MOSCAD RTU Update Interval has been changed the **POLL_NOW** item can be used.

Several MDLCGATE Servers connected to one Gateway.

It is possible to connect to one Gateway from two or more MDLCGATE Servers running on different PCs. In this case several things has to be observed:

- 1) If Servers are communicating with same Gateway simultaneously then **Max Pending Read Requests** and **Max Pending Write Requests** parameters must be set. These parameters are set in the (see chapter **MDLCGATE Gateway Node Settings**) on each PC where Server is running. These parameters should be set according the following rule: the sum of **Max Pending Read Requests** for all connected Servers cannot exceed allowed capacity of pending poll requests of Gateway and the sum of **Max Pending Write Requests** cannot exceed allowed capacity of pending write requests of Gateway.
- 2) If Servers are communicating with Gateway not simultaneously (client performs swapping from one Server to another) then **Max Pending Read Requests** can be up to maximum of Gateways capacity of pending read requests and **Max Pending Write Requests** can be up to Gateways capacity of pending write requests for each Server. The Server is not resetting queue of requests when connection with Gateway by some reason is lost. This allows to process replies after reconnecting. This feature usually is not working during Servers swapping. When swapping is performed then queue of requests of just connected Server must be reset because Server waits forever for reply on sent request. The client (e.g. InTouch) must write value 1 to GATEWAYS

topic special item **RESET_QUEUE_xxx.xxx.xxx.xxx** or write value 1 to GATEWAYS
topic special items **RESET_READS_xxx.xxx.xxx.xxx** and
RESET_WRITES_xxx.xxx.xxx.xxx. Reset ensures that queue of requests will be
empty and Server can send the total amount of poll (**Max Pending Read Requests** or
default value 20) and write requests (**Max Pending Write Requests** or default value
5).

Item (Point) Naming

The MDLC Gateway Server item/point naming is based on information contained in the RTU Types Definition file. The item/point name generally may be described as:

TzCxRy[TSn] [.b][h]

where:

z - table number;

x - number of column (starting from 0) in the table;

y - number of row (starting from 0) in the table;

TSn - used only if data type is **Timestamp1** and **Timestamp2** (n can be 1 or 2); for these items the corresponding element type in the RTU Types Definition file must be **float**;

.b - optionally used address of bit if data type is Compressed Bits and only one bit must be accessed;

h - (or **H**) optionally used suffix indicating that this item value will be stored in the historical file.

Examples:

T1C2R0 - Column 2 Row 0 in Table 1;

T1C2R5h - Column 2 Row 5 in Table 1, values will be stored in the historical file;

T12C0R1.4 - bit 4 in Column 0 Row 1 in Table 12.

T12C0R1.4H - bit 4 in Column 0 Row 1 in Table 12, values will be stored in the historical file.

T1C2R0TS1 - Column 2 Row 0 in Table 1, data type is Timestamp1;

The following item types are supported.

Item Type	Value Range	Type
Bit	0,1	Discrete
Bit in Compressed Bits	0,1	Discrete
Compressed Bits (represents a Row of up to 8 Bits in a Table)	0...255	Integer
Analog	-32768...32767	Integer
Float	Standard IEEE 32 bit floating value	Real
TS1 Timestamp1	11 bytes long string	Message
TS2 Timestamp2	9 bytes long string	Message

Notes:

1. The **Type** is the item/point type used in the Wonderware InTouch.

2. The **Bit in Compressed Bits**, **TS1** and **TS2** are Read Only, all other types are Read and Write.

3. The **TS1** items have the format: "**yy/mm/dd/hh**" where **yy** is year (00...99), **mm** is month (01...12), **dd** is day (01...31) and **hh** is hour(00...23).

4. The **TS2** items have the format: "**MM:SS:MSC**" where **MM** is minutes (00...59), **SS** is seconds (00...59) and **MSC** is milliseconds(000...999).

The additional items/points can be used for special actions:

1. The special topic **GATEWAYS** items/points (**xxx.xxx.xxx.xxx** is the Gateway IP address):

QUEUE_IS_FULL_xxx.xxx.xxx.xxx - Discrete, Read Only; used for indication of polling queue overflow; this item value is 0 if the current number of poll requests for active Gateway does not exceed **Max Pending Read Requests**(see chapter **MDLCGATE Gateway Node Settings**) ; this item value changes to 1 if there are more than **Max Pending Read Requests** for this Gateway (see also **Guidelines on Server Performance** section).

RESET_QUEUE_xxx.xxx.xxx.xxx - Discrete, Read-Write; used for reset of polling and writing queue overflow; If this item value is changed from 0 to 1 then both read and write queue will be reseted. After setting "RESET_QUEUE_xxx.xxx.xxx.xxx" to 1 replies on previous requests are not expected. The capacity of write requests queue becomes equal to **Max Pending Write Requests**(see chapter **MDLCGATE Gateway Node Settings**) and capacity of read(poll) requests queue becomes equal to **Max Pending Read Requests**. Value can be switched to 1 and both queues will be emptied. In case of redundant Gateway use only Primary Gateway address in the item name because **RESET_QUEUE_xxx.xxx.xxx.xxx** item belongs to MDLCGATE Gateway node (see also **Guidelines on Server Performance** section).

READ_COUNTER_xxx.xxx.xxx.xxx - Integer, Read Only; used for monitoring of read(poll) message queue; initial value of this item is **Max Pending Read Requests** (see chapter **MDLCGATE Gateway Node Settings**). Value decrease by each read request sending and increase by answer receiving. Value of item represents how many reads can be sent to active Gateway. In the reads included active Gateway STATUS message. For Gateway STATUS message one position is reserved because it must be sent in its update interval. It means that actually amount of reads is **Max Pending Read Requests - 1**. If value of item becomes 1 only Gateway STATUS message can be sent. In this case the warning message "Read(Poll) Message Queue is FULL. Only GW STATUS can be requested" appears in the VWLogger. If value of item is 0 read(poll) queue is full and server waits for answer on sent requests. In case of redundant Gateway use only Primary Gateway address in the item name because **READ_COUNTER_xxx.xxx.xxx.xxx** item belongs to MDLCGATE Gateway node.

WRITE_COUNTER_xxx.xxx.xxx.xxx - Integer, Read Only; used for monitoring of write message queue; initial value of this item is **Max Pending Write Requests** (see chapter **MDLCGATE Gateway Node Settings**). Value decreases by each write request sending and increase by answer receiving. Value of item represents how many writes can be sent to Gateway. If value of item is 0 write queue is full and server waits for answer on sent requests. In this case the warning message "Write Pending Message Queue is FULL." appears in the VWLogger. In case of redundant Gateway use only Primary Gateway address in the item name because **WRITE_COUNTER_xxx.xxx.xxx.xxx** item belongs to MDLCGATE Gateway node.

GS_LSB_xxx.xxx.xxx.xxx - Integer, Read Only; used for storing of two LSBytes with Gateway mode indications.

GS_MSB_xxx.xxx.xxx.xxx - Integer, Read Only; used for storing of two MSBytes with Error and Warning indications.

GMODE_SET_xxx.xxx.xxx.xxx - Discrete, used for setting of the Gateway operation mode - when switched from 0 to 1 then MDLC Gateway API function *gwlib_set_mode()* is called with *mode* set to *GWLIB_PRIMARY_MODE*; if switched from 1 to 0 then called with *mode* set to *GWLIB_SECONDARY_MODE*.

SEND_ERROR_xxx.xxx.xxx.xxx - Integer, Read Only; used for storing of return values from MDLC Gateway API functions *gwlib_poll()*, *gwlib_send_data()*, *gwlib_send_command()*, *gwlib_set_time()*, *gwlib_get_time()*;

RECEIVE_ERROR_xxx.xxx.xxx.xxx - Integer, Read Only; used for storing of return values from MDLC Gateway API functions *gwlib_receive_buffer()*, *gwlib_get_message()*.

DUMP – Discrete; all information about gateway nodes, topics, messages and data items are logged. This can be used to find out how many messages are actually sent to the network.

Example:

GS_LSB_192.17.33.160 - Integer, Read Only; used for storing of Gateway mode indication for the Gateway with IP address 192.17.33.160.

2. The special items/points which can be created for each topic (RTU):

POLL_NOW - Discrete, used for immediately one time polling of all active items (see *Guidelines on Server Performance* section); this one time polling is performed when **POLL_NOW** value switches from 0 to 1; no effect when switches from 1 to 0;

UPDATE_INTERVAL - Integer, used for run-time changing of *MOSCAD RTU Update Interval* for this Topic (see *Guidelines on Server Performance* section);

SET_TIME - Discrete, used for downloading of Server date and time (by calling *gwlib_set_time*) to a specified RTU;

GET_TIME - Discrete, used for uploading of a specified RTU date and time; the date and time request (by calling MDLC Gateway API function *gwlib_get_time*) is sent when **GET_TIME** value switches from 0 to 1; no effect when switches from 1 to 0; the result is stored in the item **DATE_TIME**;

DATE_TIME - Message, Read Only; used for storing of character string containing the specified RTU date and time in the following format: "YY/MM/DD hh:mm:ss" where YY is year, MM - month, DD - day, hh - hours, mm - minutes and ss - seconds;

SUSPEND - Discrete, Read Write; used for suspending of poll messages of topic; value 0 allows poll message sending, but value 1 disables all current polls and only write messages can be sent; poll can be executed to each topic by **POLL_NOW** = 1 (see also *Guidelines on Server Performance* section);

Historical File

Only data marked with suffix h(or H) are stored in the historical file. This file is updated at the historical data update interval. The name of historical file is *Hyyymmdd.LOG* where *yy* is year, *mm* - month and *dd* - day. For example, H960702.LOG is the name of a historical file with data received from all connected Gateways at July 2, 1996.

This single file record (row) contains a single received item data. The format of this single file record is as follows:

yy/mm/dd,HH:MM:SS:MSEC,ID,STAT,TzCxRy,value

where: **yy** is year(00...99), **mm** - month (0...11), **dd** - day(1...31), **HH** - hours (00...23),
MM - minutes (00...59), **SS** - seconds (00...59), **MSEC** - milliseconds (000...999);
ID - RTU Site ID;
STAT - the RTU Status,
TzCxRy - the address of item (table, column and row),
value - received value.

If the same table contains TS1 and TS2 elements then RTU data and time are stored; otherwise the computer data and time are stored.

The following would be an example of one this file record if RTU Site ID is 13, RTU Status is 0, item address is Table 2 Column 5 Row 0 and received data value is 1:

96/02/07,09:48:40:760,13,0,T2C5R0,1

Using the MDLCGATE Server with InTouch

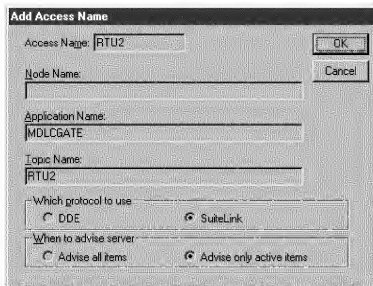
To access items/points on the MOSCAD RTUs from **InTouch**, the Access names and Tag names should be defined in **WindowMaker**.

Defining the DDE Access names

To define the Access Names in WindowMaker for each node invoke the */Special/Access Names...* command. The "Access Names" dialog box will appear.



Click on **Add**. The "Add Access Name" Dialog Box will appear:



Note. If **Add** is selected, this dialogue box will be blank when it initially appears. Data has been entered here to illustrate the entries that are made.

The following fields are required entries when entering an Access Name Definition:

Access Name

Enter an arbitrary name that will be used by **InTouch** to refer to the topic. For simplicity, it is recommended that the name defined for the topic in MDLCGATE Server also be to be used here.

Node Name

If the data resides in a network I/O Server, in the Node Name box, type the remote node's name.

Application Name

In the Application Name box, type the actual program name for the I/O Server program from which the data values will be acquired. In case the values are coming from the MDLCGATE Server the MDLCGATE is used. Do not enter the .exe extension portion of the program name.

Topic Name

Enter the name defined for the topic in the MDLCGATE Server to identify the topic the MDLCGATE Server will be accessing. The Topic Name is an application-specific subgroup of data elements. In the case of data coming from a MDLCGATE Server program, the topic name is the exact same name configured for the topic in the MDLCGATE Server.

Note: *This will usually be the same as the "Access Name", although, if desired, they may be different. However, it must be the same name used when the topics were configured in section **Configuring the MDLCGATE Server**.*

Which protocol to use

Select the protocol (**DDE** or **SuiteLink**) that you are using.

When to advise server

Select **Advise all items** if you want the Server program to poll for all data whether or not it is in visible windows, alarmed, logged, trended or used in a script. Selecting this option will impact performance, therefore its use is not recommended.

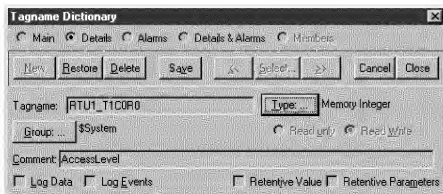
Select **Advise only active items** if you want the Server program to poll only points in visible windows and points that are alarmed, logged, trended or used in any script.

Click **OK** to accept the new Access Name and close the "Add Access Name" dialogue box. The "Access Names" dialogue box will reappear displaying the new Access Name selected in the list.

Click **Close** to close the "Access Names" dialogue box.

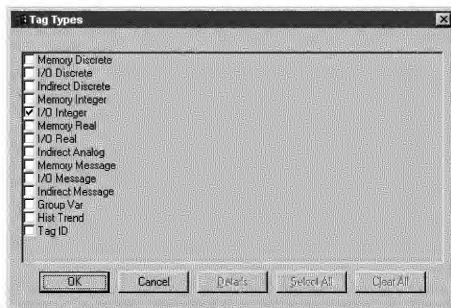
Defining the Tag Names

To define the Tag names associated with the new "Access Name", invoke the */Special/Tagname Dictionary...* command (in **WindowMaker**). The "Tagname Dictionary" dialogue box will appear:



Click on **New** and enter the **Tagname**. (The tagname defined here is the name InTouch will use. The MDLCGATE Server does not see this name.)

Select the tag type by clicking on the **Type: ...** button. The "Tag Types" dialogue box will appear:



To access MDLCGATE device items, the type must be **I/O Discrete**, **I/O Integer** or **I/O Message**. Select the Tag type.

The "Details" dialogue box for the tagname will appear:

Initial Value:	0	Min EU:	-32768	Max EU:	32767
Deadband:	0	Min Raw:	-32768	Max Raw:	32767
Eng Units:					Conversion: <input checked="" type="radio"/> Linear <input type="radio"/> Square Root
Access Name: ...		Unassigned			
Item:					
<input type="checkbox"/> Use Tagname as Item Name					Log Deadband: 0

Select the Access name for MDLCGATE Server by clicking on the **Access Name: ...** button. The "Access Names" dialogue box will appear:

Access Names	
RTU1	Done
	Add...
	Modify...
	Delete

Select the appropriate Access Name and click on **Close**. (If the Access Name has not been defined as previously described, click on **Add...** and define the Access Name now.)

The "Details" dialogue box will appear displaying the selected Access Name:

Initial Value:	0	Min EU:	-32768	Max EU:	32767
Deadband:	0	Min Raw:	-32768	Max Raw:	32767
Eng Units:					Conversion: <input checked="" type="radio"/> Linear <input type="radio"/> Square Root
Access Name: ...		RTU1			
Item:					
<input type="checkbox"/> Use Tagname as Item Name					Log Deadband: 0

For integers fill in the **Min EU**, **Max EU**, **Min Raw** and **Max Raw** fields. These fields control the range of values that will be accepted from the Server and how the values are scaled. If no scaling is desired, **Min EU** should be equal to **Min Raw** and **Max EU** equal to **Max Raw**.

Enter the MDLCGATE item name to be associated with this tagname in the Item: field in the "Details" box:

Initial Value:	0	Min EU:	-32768	Max EU:	32767
Deadband:	0	Min Raw:	-32768	Max Raw:	32767
Eng Units:					Conversion: <input checked="" type="radio"/> Linear <input type="radio"/> Square Root
Access Name:	RTU1				
Item:	T1COP0				
<input checked="" type="checkbox"/> Use Tagname as Item Name					Log Deadband: 0

(Refer to the **Item Names** section for complete details.)

Where applicable, the **Use Tagname as Item Name** option may be selected to enter automatically the tag name in this field.

Note: The tag name can only be used if it follows the conventions listed in the **Item Names** section.

Once all entries have been made, click on the **Save** button (in the top dialogue box) to accept the new tag name. To define additional tagnames click on the **New** button. To return to the **WindowMaker** main screen, select **Close**.

Monitoring the Status of Communication with InTouch

InTouch supports built-in topic names called **DDEStatus** and **IOStatus** that are used to monitor the status of communications between the Server and InTouch. For more information on the built-in topic names DDEStatus and IOStatus, see your online "InTouch User's Guide".

The status of communication between the Server and InTouch can be read into **Excel** by entering the following DDE reference formula in a cell on a spreadsheet (in following examples **RTU1** is the Topic Name configured for MDLCGATE Server):

=view|DDEStatus!RTU1

or

=view|IOStatus!RTU1

Notes on Using Microsoft Excel

Data from MDLCGATE topics (RTUs) may be accessed from Excel spreadsheets. To do so, enter a formula like the following into a cell on the spreadsheet.

=MDLCGATE|topic|item

Sometimes, Excel requires the topic and/or item/points to be surrounded by apostrophes.

In the formula, **topic** must be replaced with one of the valid topic names defined during the Server configuration process. Replace **item** with one of the valid item/point names described in the **Item (Point) Naming** section.

Reading Values into Excel Spreadsheets

Values may be read directly into Excel spreadsheets by entering a DDE formatted formula into a cell, as shown in the following examples:

=MDLCGATE|'RTU2'|' T1C2R0'
=MDLCGATE|' RTU3'|' T2C5R1.3'
=MDLCGATE|' RTU10'|' T1C2R0TS2'

Note: Refer to the Microsoft Excel manual for complete details on entering Remote Reference formulas for cells.

Writing Values to MDLCGATE Points

Values may be written to the Server from Microsoft Excel by creating an Excel macro that uses the **POKE** command. The proper command is entered in Excel as follows:

channel=INITIATE("MDLCGATE","topicname")
=POKE(channel,"itemname", Data_Reference)
=TERMINATE (channel)
=RETURN()

The following describes each of the above **POKE** macro statements:

channel=INITIATE("MDLCGATE","topicname")

Opens a channel to a specific topic name (defined in the Server) in an application with name MDLCGATE (the executable name less the .EXE) and assigns the number of that opened channel to **channel**.

Note: By using the **channel=INITIATE** statement the word **channel** must be used in the **=POKE** statement instead of the actual cell reference. The **"applicationname"** and **"topicname"** portions of the formula must be enclosed in quotation marks.

=POKE(channel,"itemname", Data_Reference)

POKEs the value contained in the **Data_Reference** to the specific operand in the MDLCGATE via the **channel** number returned by the previously executed **INITIATE** function. **Data_Reference** is the row/column ID of the cell containing the data value. For **"itemname"**, use some of the valid item names specified like described in the **Item Names** section.

=TERMINATE(channel)

Closes the channel at the end of the macro. Some applications have a limited number of channels. Therefore they should be closed when finished. **Channel** is the channel number returned by the previously executed **INITIATE** function.

=RETURN()

Marks the end of the macro.

The following is an example of Excel macro used to poke value from cell B2 to topic **RTU1** item **T1C0R0**:

```
PokeMacro -Ctrl a
=INITIATE("MDLCGATE","RTU1")
=POKE(A2,"T1C0R0",B2)
=ON.TIME(NOW()+0.01,"TerminateDDEChannel")
=RETURN()
```

```
TerminateDDEChannel
=TERMINATE(A2)
=RETURN()
```

Note: Refer to the Microsoft Excel manual for complete details on entering Remote Reference formulas for cells.

Troubleshooting

WIN.INI entries

The first time you run the MDLCGATE Server configuration, most of the items in the following list will automatically appear in the WIN.INI file. It is usually in the C:\WINDOWS directory. It is an ASCII file and can be altered manually if you wish with any text editor, e.g., MS Windows Notepad (*do not use a program that formats text, such as MS Word or Write unless the file is saved as DOS text*). The following is a typical entry for the MDLCGATE Server.

[MDLCGATE]	
ProtocolTimer=50	Protocol Timer Tick
ConfigurationFile=C:\MDLCGATE\	Configuration File Path
WinIconic=0	Server starts up minimized (if 1)
WinFullScreen=0	Server starts up maximized (if 1)
WinTop=112	Position and size of Server's window at
	Server startup (if not minimized or
	maximized)
WinLeft=0	
WinWidth=200	
WinHeight=168	
DebugMenu=1	Show Troubleshooting menu (if 1)
ShowSend=0	ShowSend is checked (if 1)
ShowReceive=0	ShowReceive is checked (if 1)
ShowErrors=1	ShowErrors is checked (if 1)
DumpScreen=0	DumpScreen is checked (if 1)

Following optional entries for MDLCGATE Server can be used if default values of parameters must be changed:

DelayAfterLostRequest=10

Time in seconds of delay after LOST REQUEST message received from Gateway till next request sending. Value can be from 1 to 65535 seconds.

LogSend=1

The possibility to configure which subitem of the "Log Send" system menu item will be selected at start-up. Subitems has the following values:

"Log All Data" = 1;

"Custom Log" = 2;

"No Logging" = 0.

At Server first start-up the default selection is "No Logging".

LogReceive=1

The possibility to configure which subitem of the "Log Receive" system menu item will be selected at start-up. Subitems has the following values:

"Log All Data" = 1;

"Custom Log" = 2;

"No Logging" = 0.

At Server first start-up the default selection is "No Logging".

MultiWrite=0

Can be used in cases when write command execution order is important for user application the Server must be configured to transfer only one non-answered write message to each topic.

Value 0 enables only one non-answered write sending to each topic. In this case

Max Pending Write Request

parameter(see chapter **MDLCGATE Gateway Node Settings**) value is disabled. Default value of MultiWrite is 1.

WarningOnPollMiss=1

Can be used in cases when it is necessary to see that general poll issue is missed. Parameter has two valid values:

0 - Off;

1 - On.

Default parameter value is 0. If above described situation presents then warning message should appear in the WWLogger file. For example:

98/07/09 09:30:00.227/MDLCGATE/GW node:Node2 Topic:RTU01 Table:1.

Missed Poll Issue during Update Interval(900 seconds)**WarningOnPollDelay=5**

Can be used in cases when it is necessary to see that general poll issue is not missed yet but only delayed some time. Valid values of parameter are between 0 and 86400 seconds. No message will be logged if value is 0. Default this parameter value is 0. If above string is included then warning message should appear in the WWLogger file. For example:

98/07/09 09:15:05.215/MDLCGATE/GW node:Node2, Topic:RTU01, Table:1. Poll issue delay is 5 seconds.

DelayBetweenPolls=600

The possibility to configure special poll mode. Value can be from 0 to 86400 seconds. Value 0 of this parameter means that general polling mode is used and poll execution depends on update interval. Non-zero value defines time interval in seconds between STARTINGS of each consecutively processed topic polling (not between last request of currently processed topic and the first request of the next topic). Topics are polled consecutively (one by one) in the order of the initiation. If parameter **DelayBetweenPolls** appears in the MDLCGATE section of the WIN.INI file and its value differs from 0 then following changes in the Server performance are expected:

- 1) All topics are suspended. Value of SUSPEND item for each topic is equal to 1 and cannot be changed.
- 2) MOSCAD RTU Update Interval is disabled.
- 3) Ping Update Interval is disabled. The "Pings to RTU" are not executed.

Writes are executed in general way.

FirstBadTryMsgBox=0

The possibility to disable the first connection alert message box. Default value is 1.

ResetMode=0

If this parameter value is 0 the pending requests queue will be emptied when no connection to Gateways happens. Default value is 1, ie, MDLCGATE Server remembers pending request numbers till the Server shutdown. Only by reset command can clear pending(read and write) requests queue(see chapter **Item (Point) Naming**).

Troubleshooting menu

The following debugging choices are appended to the Server's System Menu (the menu that appears when you click the "-" box in the upper left hand corner of the Server window):

- Suspend Protocol/Resume Protocol** - these choices permit you to turn protocol processing on and off, which means that you can suspend access to the RTUs.
- Log Send** - this system menu item is provided for organizing of send package logging to the WWLogger file. This item has the following subitems: **"Log All Data", "Custom Log", "No Logging"**.
- Log All Data** - if checked then all outgoing user data is logged to the WWLogger file. Each message's log contains 1 or more strings. First string format differs from each next ones of message's log. First string of message's log has the following format(date, time and application name are skipped):

<type of request>:<node name> < contents of request> size:<size of request> mailbox:<#>- <request in hexadecimal format>

where

<type of request> - "Poll" or "Write" types of request are possible
<node name> - name of sender node
<contents of request>:

1. For Gateway status poll message - in the following format:
 <GW <#> STATUS>,
 where # is 'I' for PRIMARY 'II' for SECONDARY Gateway.
2. For Poll or Write requests format is the following:
 <<Topic Name> Table:<#> StartRow:<#> EndRow:<#>
 StartCol:<#> EndCol:<#>>. Request contains the Name of sender Topic and the RTU table number. The StartRow and EndRow shows the smallest and the greatest row numbers of active items of the table of the topic. The StartCol and EndCol shows the smallest and the greatest column numbers of active items of the table of the topic.

3. For "PING" of topic poll message in the following format:
<<Topic Name> PING>, where
<Topic Name> - name of sender topic.

<size of request> - size of request in bytes
<#> - number of mailbox for request and reply
<request in hexadecimal format> - hexadecimal queue of the request

For example, the Gateway Status request number 144 for node Node2:

Poll(mailbox:144):Node2 GW I STATUS(size:36):00 01 7F FD 00 00...

Each next string of log (if such is necessary) has the following format:

<type of request>+<request in hexadecimal format>

where

<type of request> - "Poll" or "Write" type (same as in the first string of the log)

<request in hexadecimal format> - continuation of the hexadecimal queue of the request

For example:

Poll+:00 00 00 00...

Custom Log - messages will be logged to the WWLogger corresponding to settings in the Server's main window;

No Logging - no messages will be logged to the WWLogger;

When the Server starts the default selection is "No Logging". This default value can be changed by settings in the WIN.INI file. Subitems has the following values:

"Log All Data" = 1;

"Custom Log" = 2;

"No Logging" = 0.

The following is an example for "Log All Data" subitem selection in the Server's system menu:

[MDLCGATE]

LogSend=1

Show Receive - this system menu item is provided for organizing of received package logging to the WWLogger file. This item has the following subitems: "Log All Data", "Custom Log", "No Logging".

Log All Data - if checked then all incoming user data is logged to the WWLogger file. Each message's log contains 1 or more strings. First string format differs from each next ones of message's log. First string of message's log has the following format:

<type of message>:<node name> size:<size of message>-< contents of message> [mailbox:<#>]- <message in hexadecimal format>

where

<type of message> - "Receive" or "Burst" types of message are possible

<node name> - name of receiver node

<size of request> - size of received message in bytes

<contents of message>:

1. Reply (acknowledge) on the Gateway status or mode poll contains the following string: "GET_MODE_ACK or GET_STATUS_ACK".
2. Reply (acknowledge) format on Poll, PING, Set time, Get time, Write and request is the following:
 <<Topic Name> <acknowledge type>>. Request contains the name of sender Topic and type of acknowledge ("POLL_ACK", "GET_TIME_ACK", "SET_TIME_ACK", "COMMAND_ACK").
3. For "Burst"(unsolicited) message in the following format:
 <<Topic Name> BURST_MSG>, where
 <Topic Name> - the name of sender topic.

<(mailbox:<#>)> - the number of mailbox for "Receive" type of message. This parameter is not used for "Burst" messages

<request in hexadecimal format> - the hexadecimal queue of the request

Custom Log - messages will be logged to the WWLogger corresponding to settings in the Server's main window.

No Logging - no messages will be logged to the WWLogger.

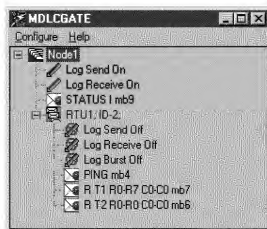
At Server first start-up the default selection is "No Logging". This default value can be changed by settings in the WIN.INI file. Subitems has the following values:

"Log All Data" = 1;
 "Custom Log" = 2;
 "No Logging" = 0.

The following is an example for "Log All Data" subitem selection in the Server's system menu.

```
[MDLCGATE]
LogReceive=1
```


- Show Errors** - if checked then all information about errors is logged.
- Dump** - all information about gateway nodes, topics, messages and data items are logged. This can be used to find out how many messages are actually sent to the network.
- Dump Screen** - if checked then information about open connections and active messages is displayed as a tree of communication component structure and it is drawn in the Server's main window.



where

- "Gateways Nodes" and special topic "Gateways"
- Initialized Topic Name and Topic ID
- Log On. Shows which type of data ("Send", "Receive", "Burst") will be logged
- Log Off. Shows which type of data ("Send", "Receive", "Burst") will be not logged

"Log On" or "Log Off" mode can be set by mouse left button double click on the item. These settings are in use if "Custom Log" menu subitem of "Log Send"/"Log Receive" menu item is selected in the Server's system menu.

- Requests:
 - "STATUS I" - item shows "Gateway Status Poll Message" with it's last mailbox number.
 - "STATUS II" - item shows "Redundant Gateway Status Poll Message" (if redundant Gateway is configured) with it's last mailbox number.
 - "Ping" - item shows Topic Ping Status message with its last mailbox number.
 - "Read Message" - item shows general read message in the following format:

R T<#> R<s#>-R<e#> C<s#>-C<e#> mb<#>

Where

- R - letter representing the read message;
- T<#> - table number (from Rtutypes.cfg);
- R<s#> - from which row the message starts (start row);
- R<e#> - which row is the last in the message (end row);
- C<s#> - from which column (from Rtutypes.cfg) the message starts (start column);
- C<e#> - which column (from Rtutypes.cfg) is the last in the message (end column);

mb<#> - last mailbox number of poll (read) message.

Note: Read message disappears when the last item is unadvised.

"Write Message" - item shows general write message in the following format:

W T<#> R<s#>-R<e#> C<s#>-C<e#> mb<#>

Where

W - letter representing the write message;

T<#> - table number (from Rtutypes.cfg);

R<s#> - from which row the message starts (start row);

R<e#> - which row is last in the message (end row);

C<s#> - from which column (from Rtutypes.cfg) the message starts (start column);

C<e#> - which column (from Rtutypes.cfg) is last in the message (end column);

mb<#> - last mailbox number of poll (read) message.

Note: Write message disappears when reply is received.

By mouse right button clicking it is possible to access the submenu. The submenu contains the following items:

Log All Sends - selects all "Log Send" treeview items of topics of the current gateway node.

Log All Receives - selects all "Log Receive" treeview items of topics of the current gateway node.

Log All Bursts - selects all "Log Burst" treeview items of topics of the current gateway node.

No Log Sends - unselects all "Log Send" treeview items of topics of the current gateway node.

No Log Receives - unselects all "Log Receive" treeview items of topics of the current gateway node.

No Log Bursts - unselects all "Log Burst" treeview items of topics of the current gateway node.

Dump - dump internal data of the Server to WWLogger. Data will be logged depending on treeview item that is under mouse cursor (the clicking on topic initiates the internal info logging for this topic).

All debug information (except **Dump Screen**) is displayed via the Wonderware Logger, which must be active for these commands to work.

Warning: if you check **Log Send/Log All Data** and/or **Log Receive/Log All Data** debug output grows very fast and it is possible that computer becomes very slow.

MOSCAD System Definition Files

The definition of the entire MOSCAD System (all RTUs for all Gateway Nodes) is stored in the two files:

1. RTU Types Definition file.
2. MOSCAD System Definition file.

RTU Types Definition

In the RTU Types Definition file the RTU TYPES in the MOSCAD System are defined.

Each RTU Type Definition includes:

- I. Number of Tables
- II. For each Table:
 - A. Table Number (in the MOSCAD application)
 - B. Number of columns in the Table
 - C. Type of each column

The file is case insensitive.

```
-----  
; File Type & Version  
-----  
;   
*File_Type =RTU_TYPE  
*Version=XXX  
-----  
;   
; Definition of Types ;  
-----  
* No_of_Types=Nt  
;   
;   
-----  
; Definition for Type No ;  
-----  
*Type_name No_of_tables  
;   
;   
-----  
; Definition for Table No j. in Type No ;  
-----  
* Table_no No_of_columns Column_type_0 ... Column_type_i
```

Example:

```
;RTU Types File Version
FILE_TYPE RTU_TYPE
VERSION 1.0
```

```
; Definition of RTU Types
Number_of_Types 3
```

```
;
;
;
;      Type_name No_of_tables
Type Pump 3
;
;      Table_no N_cols Col_0 Col_1 Col_2 Col_3 Col_4 Col_5 Col_6 Col_7
Table 0      5      Bit   Bit   Bit   Bit   Bit
Table 2      8      C_Bit C_Bit C_Bit C_Bit C_Bit C_Bit C_Bit C_Bit
Table 3      3      Float Val   Bit
;
;
;      Type_name No_of_tables
Type Valve 1
;
;
;      Table_no N_cols Col_0 Col_1 Col_2 Col_3 Col_4 Col_5 Col_6 Col_7
Table 10     3      Float Val   Bit
;
;      Type_name No_of_tables
;
;
Type PoleTop 1
;
;
;      Table_no N_cols Col_0 Col_1 Col_2 Col_3 Col_4 Col_5 Col_6 Col_7
Table 1      1      Bit
```

MOSCAD System Definition

In the MOSCAD System Definition file the association of RTU - 'RTU Type' is defined. Each RTU (defined by its Site_id) is assigned an 'RTU Type' (defined in the RTU Types Definition file).

The file is case insensitive.

```

;-----
;File Version
;-----
*File_Type=SYSTEM
*Version=XXX
;-----
;Definition of RTUs      ;
;-----
*No_of_Rtus
;
;
;-----
;Definition for RTU No. i ;
;-----
*   Site_id   Type_name

```

Example:

```

;MOSCAD System File Version
FILE_TYPE SYSTEM
VERSION 1.0
;   No of RTUs
N_RTU 4
;
;
;   Site_id   Type_name
RTU 1 Pump
RTU 2 Pump
RTU 3 PoleTop
RTU 4 Valve

```

KLINKMANN AUTOMATION
MDLC Gateway DDE Server
Revision History

Sep 96	Rev 1.0	First Release
Nov 96	Rev 1.1	Hot Standby - Principles of Operation section added. Guidelines on Server Performance section added. Installing and starting the MDLCGATE Server section modified. Gateway Node Definition Command section modified. Topic Definition Command section modified. Historical File section modified. Item (Point) Naming section modified.
Dec 96	Rev 1.2	
Feb 97	Rev 1.3	
Sep 97	Rev 1.4	Manual file name changed. Minor changes.
Oct 97	Rev 1.5	Changes in Item(Point) Names section added. Topic Definition Command section modified. Troubleshooting Menu section modified.
Oct 98	Rev 1.6	Troubleshooting section modified. Topic Definition Command section modified. Guidelines on server performance section modified. Item(point) naming section modified.
Mar 2000	Rev 1.7	Item(point) naming section modified. Gateway Node Definition Command section modified. Communication Protocols section modified. Other minor changes.
Apr 2000	Rev 1.8	Suite Link support added. Max Pending Read and Write Requests added in MDLCGATE Gateway Node Settings" dialog.
Mar 2002	Rev 1.9	Installation from CD information added.

X. RELATED PROCEEDINGS

None.